我觀看你指頭所造的天、並你所陳設的月亮星宿、便說、人算甚麼、你竟顧念他．世人算甚麼、你竟眷顧他。

When I consider your heavens, the work of your fingers, the moon and the stars, which you have set in place, what is man that you are mindful of him, the son of man that you care for him?

e-Design
Computer-Aided Engineering Design

Kuang-Hua Chang

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO
Academic Press is an imprint of Elsevier

Academic Press is an imprint of Elsevier
125 London Wall, London EC2Y 5AS, UK
525 B Street, Suite 1800, San Diego, CA 92101-4495, USA
225 Wyman Street, Waltham, MA 02451, USA
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK

Copyright © 2015 Elsevier Inc. All rights reserved.

This book and the individual contributions contained in it are protected under copyright by the Publisher (other than as may be noted herein).

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without permission in writing from the publisher. Details on how to seek permission, further information about the Publisher's permissions policies and our arrangements with organizations such as the Copyright Clearance Center and the Copyright Licensing Agency, can be found at our website: www.elsevier.com/permissions.

Notices
Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

ISBN: 978-0-12-382038-9

For information on all Academic Press publications visit our website at http://store.elsevier.com/

Publisher: Joe Hayton
Acquisition Editor: Steve Merken
Editorial Project Manager: Chelsea Johnston
Production Project Manager: Lisa Jones
Designer: Matthew Limbert

Typeset by TNQ Books and Journals
www.tnq.co.in

Printed and bound in the United States of America

Preface

The conventional product development process employs a design–build–test philosophy. The sequentially executed product development process often results in a prolonged lead time and an elevated product cost. The e-Design paradigm presented in this book employs IT-enabled technology, including computer-aided design, engineering, and manufacturing (CAD/CAE/CAM) tools, as well as advanced prototyping technology to support product design from concept to detailed designs, and ultimately manufacturing. This e-Design approach employs virtual prototyping (VP) technology to support a cross-functional team in analyzing product performance, reliability, and manufacturing costs early in the product development stage and in conducting quantitative trade-offs for design decision making. Physical prototypes of the product design are then produced using rapid prototyping (RP) technique mainly for design verification. The e-Design approach holds potential for shortening the overall product development cycle, improving product quality, and reducing product cost. This book intends to provide readers with a comprehensive coverage of essential elements for understanding and practicing the e-Design paradigm in support of product design, including design method and process, and computer-based tools and technology. The book consists of four parts: Product Design Modeling, Product Performance Evaluation, Product Manufacturing and Cost Estimating, and Design Theory and Methods. The Product Design Modeling discusses virtual mockup of the product that is first created in the CAD environment. The critical design parameterization that converts the product solid model into parametric representation, enabling the search for better designs, is an indispensable element of practicing the e-Design paradigm, especially in the detailed design stage. The second part, Product Performance Evaluation, focuses on applying computer-aided engineering (CAE) technology and software tools to support evaluation of product performance, including structural analysis, fatigue and fracture, rigid body kinematics and dynamics, and failure probability prediction and reliability analysis. The third part, Product Manufacturing and Cost Estimating, introduces computer-aided manufacturing (CAM) technology to support manufacturing simulations and process planning, RP technology, sheet-metal forming, and computer numerical control (CNC) machining for fast product prototyping, as well as manufacturing cost estimate that can be incorporated into product cost calculations. The product performance, reliability, and cost calculated can then be brought together to the cross-functional team for design trade-offs based on quantitative engineering data obtained from simulations. Design trade-off is one of the key topics included in the fourth part, Design Theory and Methods. In addition to conventional design optimization methods, we discuss decision theory, utility theory, and decision based design. Simple examples are included to help readers understand the fundamentals of concepts and methods introduced in this book.

In addition to the discussion on design principles, methods, and processes, this book offers detailed review on the commercial off-the-shelf software tools for the support of modeling, simulations, manufacturing, and product data management and data exchanges. Tutorial-style lessons on using commercial software tools are provided together with project-based exercises. Two suites of engineering software are covered: they are Pro/ENGINEER-based, including Pro/MECHANICA Structure, Pro/ENGINEER Mechanism Design, and Pro/MFG; and SolidWorks-based, including SolidWorks Simulation, SolidWorks Motion, and CAMWorks. In addition, Mastercam is included to enhance the learning experience in computer-aided machining simulation. These tutorial lessons are designed to help readers gain hands-on experience to practice the e-Design paradigm.

We start by providing a brief introduction to the e-Design paradigm and tool environment in Chapter 1, in which two practical examples, a simple airplane engine and a high-mobility multipurpose wheeled vehicle (HMMWV), are employed for illustration. Following this introduction, more details are offered in 18 chapters organized into four parts.

The objective of Part I, Product Design Modeling, is to provide readers with a fundamental understanding in product modeling principles and modern engineering tools for solid and assembly modeling, and apply the principles and software tools to support practical design applications. Important topics in product design modeling, including geometric and solid modeling, assembly modeling, design parameterization, and product data management and data exchange are discussed.

Chapter 2 focuses on geometric modeling, in which general geometric modeling techniques and methods commonly employed in CAD are discussed. Fundamentals in geometric modeling, such as mathematic representation of parametric curves and surfaces, continuity, and geometric transformations are presented to provide readers a basic understanding in geometric modeling. The goal of this chapter is to help readers understand how geometric entities, such as curves and surfaces, are created in CAD, which is critical to understanding the theories and methods that support part modeling in CAD.

Chapter 3 offers basic knowledge on the theories of solid modeling in CAD. Basic solid modeling theories, including constructive solid geometry (CSG), boundary representation (B-Rep), and feature-based parametric solid modeling, are briefly presented. The goal of this chapter is to help readers understand how solid parts are created in CAD and the theories and methods that support part modeling in CAD.

Chapter 4 provides a brief discussion on product assembly in CAD, which involves both modeling and analysis of the articulated assemblies for support of product design. In CAD, an assembly is created by defining relative position and orientation of parts, whereas a kinematic model is created by specifying kinematic constraints between parts. Both are important for engineers to create functional assemblies in CAD to support product design. The goal of this chapter is to help readers understand how solid parts are put together in CAD that perform desired functions and the theories and methods that do the tricks.

Chapter 5 is the key chapter of this part, in which design parameterization concept and method are discussed for the support of capturing design intents in the parts and assembly of the product model. A set of guidelines are presented for the designers to parameterize solid models at sketch, part, and assembly levels in order to properly capture design intents. The goal of the chapter is to provide design parameterization concept, methods, and guidelines that support designers to explore product design alternatives in the context of e-Design paradigm.

After learning how parts and assemblies are created in CAD, in Chapter 6 we discuss how to manage product data to support product design. In addition, data exchange between CAD systems, which is one of the major issues encountered in product design using e-Design paradigm, is discussed to offer readers practical approaches in dealing with such issues.

In addition to theories and methods, two companion projects are included: Project S1 Solid Modeling with SolidWorks and Project P1 Solid Modeling with Pro/ENGINEER. These projects offer tutorial lessons that help readers to learn and be able to use the respective software tools for support of solid modeling, assembly modeling, design parameterization, and model translations for practical applications. These tutorial lessons and example files needed for going through the lessons are available for download on the book's companion website.

Part II, Product Performance Evaluation, provides readers with fundamental understanding in product performance evaluation, which enables them to apply the principles, methods, and software tools to support practical design applications. Important topics in product performance evaluation, including structural performance of critical components, kinematics and dynamics of mechanical systems, fatigue and fracture, as well as product reliability analysis at both component and system levels, will be discussed.

Chapter 7 focuses on structural analysis, including both analytical methods and finite element analysis (FEA), in which the essential elements in using FEA for modeling and analysis of structural performance are discussed. In addition, two companion projects are included: Project S3 Structural FEA and Fatigue Analysis Using SolidWorks Simulation and Project P3 Structural FEA and Fatigue Analysis Using Pro/MECHANICA Structure. These two projects offer tutorial lessons that help readers to learn and be able to use the software tools for solving problems that are beyond hand calculations using analytical methods. The goal of this chapter is to help readers become confident and competent in using FEA for creating adequate models and obtaining reasonably accurate results to support product design.

Chapter 8 provides an overview on motion analysis. Again, both analytical and computer-aided methods, that is, the so-called computer-aided kinematic and dynamic analyses, are included. General concept and process in carrying out motion simulation for kinematic and dynamic analysis are included in this chapter. In order to support readers to use the computer-aided analysis capability for general design applications, we have provided two companion projects: Project S2 Motion Analysis Using SolidWorks Motion and Project P2 Motion Analysis Using Pro/ENGINEER Mechanism Design. Tutorial lessons of these two projects should help readers to carry out motion simulations. Again, the goal of this chapter is to help readers become confident and competent in using motion software tools for engineering design.

Chapter 9 offers a brief discussion on structural fatigue and fracture, which is one of the most technically challenging issues facing aerospace and mechanical engineers. In addition to basic theory, this chapter provides a brief review on the computational methods that support structural fatigue and fracture analysis in various stages. Similar to the previous chapters, tutorial lessons that provide details in using SolidWorks Simulation and Pro/MECHANICA Structure for crack initiation calculations are offered. You may find these lessons in Projects S3 and P3. The goal of this chapter is to enable readers to create adequate models and obtain reasonable results that support design involving fatigue and fracture.

In engineering design, there are uncertainties we must consider. Uncertainties exist in loading, material properties, geometric size, material strength, and so on. Mechanical engineers must understand the importance of the probabilistic aspect in product design and must be able to apply adequate reliability analysis methods to solve engineering problems. Chapter 10 provides a brief overview on reliability analysis, which calculates failure probability of a prescribed performance measure considering uncertainties. This chapter also touches on design from a probabilistic perspective and compares the effectiveness of the probabilistic approach with conventional methods, such as safety factor and worst-case scenario. The goal of this chapter is to provide basic probabilistic theory and reliability analysis methods that enable readers to deal with basic engineering problems involving uncertainties.

The objective of Part III, Product Manufacturing and Cost Estimating, is to provide readers with a fundamental understanding of product manufacturing principles and modern engineering tools for

manufacturing simulation and cost estimating, and to enable readers to apply principles and software tools to support practical design applications. Important topics in product manufacturing and cost estimating, including CNC machining simulation, toolpath generation, sheet metal forming simulation, rapid prototyping, and cost estimate, will be discussed.

Chapter 11 focuses on virtual machining, which is a simulation-based technology that supports engineers in defining, simulating, and visualizing the manufacturing process in a computer environment using computer-aided manufacturing (CAM) tools. In addition to virtual machining, practical aspects of CNC machining, such as fixtures, cutters, machining parameters, and CNC mill operations, are included to aid readers in bringing such considerations into machining for support of design. Three companion projects are included: Project S4: Machining Simulation Using CAMWorks, Project P4: Machining Simulation Using Pro/MFG, and Project M4: Machining Simulation Using Mastercam. These three projects offer tutorial lessons that should help readers to learn and be able to use the software tools in machining simulations for practical applications. The goal of this chapter is to help readers become confident and competent in using CAM tools for creating adequate machining simulations to support product design.

Chapter 12 provides a brief discussion of toolpath generation for surface milling, which is one of the most important machining applications. The goal of this chapter is to provide readers with a general understanding of toolpath generation, specifically for surface milling; to help readers understand the impact of machining parameters and cutters on the resulting toolpath or CL data; and to offer a detailed discussion on scallop height calculations that determine the quality of a machined surface with a quantitative measure.

Chapter 13 offers a short introduction to simulation of sheet metal forming, which is one of the most widely used manufacturing processes for thin-shell parts in the automotive and aerospace industries. In addition to basic theory, this chapter provides a brief review on the computational method that supports forming simulation as well as tooling design and process planning using simulation. Software tools commercially available for forming simulations are briefly reviewed in hope of providing readers a general idea about the availability of such tools and engineering capabilities they offer. Case studies are provided that support readers to understand practical applications of such simulation technology. The goal of this chapter is to enable readers to understand basic forming theory, create adequate simulation models and obtain reasonable results that support product design and manufacturing involving thin-shell structures.

Chapter 14 introduces Rapid Prototyping (RP), also called 3D Printing or Solid Freeform Fabrication (SFF), which is the technology and apparatus that fabricate physical objects directly from parts created in CAD using additive layer manufacturing techniques without manufacturing process planning, tooling, or fixtures. This technology has the potential to reduce the turnaround time in product design and development. The goal is to provide readers with a general understanding of RP technology and various machines commercially available, to help readers become more familiar with emerging RP and its applications in micro-manufacturing and other fields, and, through case studies, to help readers apply the same principles and methods to their own applications.

In engineering design, cost is often the driving factor that shapes the final product. The actual setting of price is at the heart of the business and is crucial to survival. Chapter 15 introduces fundamental elements in modern methods of product cost estimating. In addition, software tools for fast cost estimates in support of product design are discussed. The goal of this chapter is to help readers

understand the basics of cost estimates, employ the methods in practical applications, and acquire adequate software tools for support of design.

Part IV, Design Theory and Methods, provides readers with a fundamental understanding in product design theory and methods, and apply the theory and methods to support engineering design applications in the context of e-Design. Important topics, including decision methods and theory in engineering design, design optimization, structural design sensitivity analysis, as well as multi-objective design optimization will be discussed.

Chapter 16 focuses on decision-making for engineering design, in which conventional decision methods and decision theory, as well as decision-based design developed recently, are discussed. The conventional methods, such as decision tree and decision table, have been widely employed by industry in support of design decision-making. On the other hand, decision theory offers a scientific and theoretical basis for design decision-making, which gained the attentions of researchers in recent years. This chapter offers a short review on popular decision methods, design theory, as well as the application of the theory to support engineering design. This chapter serves as a prelude to chapters that follow in Part IV.

Chapter 17 discusses design optimization, which is one of the mainstream methods in engineering design. We discuss linear and non-linear programming and offer a mathematical basis for design problem formulation and solutions. We include both gradient-based and non-gradient approaches for solving optimization problems. In this chapter, readers should see clearly the limitations of the non-gradient approaches in terms of the computational efforts of the design problems, especially large-scale problems. The gradient-based approaches are more suitable to the typical problems in the context of e-Design. We focus on single-objective optimization that serves as a gateway to understand multi-objective optimization to be discussed in Chapter 19 that is much more relevant to practical design applications. We address issues involved in dealing with practical engineering design problems and discuss an interactive design approach, including design trade-off and what-if study, which is more suitable for support of large-scale design problems. We offer case studies to illustrate practical applications of the methods discussed and a brief review on software tools that are commercially available for support of various types of optimization problems.

Chapter 18 provides a brief discussion on the sensitivity analysis, that is, gradient calculations of product performance with respect to design variables, which are essential for design using the gradient-based methods. In this chapter, we narrow our focus on structural problems in hope of introducing basic concept and methods. We include in this chapter popular topics, such as sizing, shape, and topology designs. We also offer case studies to illustrate practical applications of the methods discussed. Some aspect of the ideas and methods on gradient calculations for structural problems can be extended to support other engineering disciplines; for example, design for mechanical motion. A case study is presented to illustrate a practical scenario that involves integration of topology and shape optimization.

In Chapter 19 we introduce multi-objective design optimization concept and methods. We start with simple examples to illustrate the concept and introduce Pareto optimality. We then discuss major solution techniques categorized by the articulation of preferences. We also include multi-objective genetic algorithms that gained popularity in recent years. In addition, we revisit decision-based design using both utility theory and game theory introduced in Chapter 16. We make a few comments on the decision-based design approach from the context of multi-objective optimization. We include a discussion on software tools that offer readers knowledge on existing tools for adoption and further

investigation. We also include two advanced topics, reliability-based design optimization and design optimization for product manufacturing cost.

In addition to theories and methods, two companion projects are included: Project S5 Design with SolidWorks and Project P5 Design with Pro/ENGINEER. We include two examples in each project, design optimization of a cantilever beam, and multi-disciplinary design optimization for a single-piston engine. The goal of the projects is to help readers become confident and competent in using CAD/CAE/CAM and optimization tools for creating adequate product design models and adopt effective solution techniques in carrying out product design tasks.

As you may notice, any individual chapters in this book could easily be expanded to a full textbook. Please keep in mind, however, that this book is not intended to provide you with detailed and thorough discussions of their respective subjects, but to offer readers the concept and process of the e-Design paradigm and the applications of computer-aided engineering technology and software tools to support modeling, simulation, and manufacturing aspects of engineering design.

This book should serve well for a two-semester (30-week) instruction in engineering colleges of general universities. Typically, a 3-hour lecture and 1-hour laboratory exercise per week are desired. This book aims at providing engineering senior and first-year graduate students a comprehensive reference to learn advanced technology in support of engineering design using IT-enabled technology. Typical engineering courses that the book serves include Engineering Design, Integrated Product and Process Development, Concurrent Engineering, Design and Manufacturing, Modern Product Design, Computer-Aided Engineering, as well as Senior Capstone Design. In addition to classroom instruction, this book should support practicing engineers who wish to learn more about the e-Design paradigm at their own pace.

RESOURCES AVAILABLE WITH THIS BOOK

For Instructors using this book for a course, an instructor manual and set of PowerPoint slides are available by registering at www.textbooks.elsevier.com. For readers of this book, in addition to the companion projects, updates and other resources related to the book, including project tutorials using ProENGINEER and SolidWorks, are available by visiting http://booksite.elsevier.com/9780123820389.

About the Author

Dr. Kuang-Hua Chang is a David Ross Boyd Professor and Williams Companies Foundation Presidential Professor at the University of Oklahoma (OU), Norman, OK. He received his diploma in Mechanical Engineering from the National Taipei Institute of Technology, Taiwan, in 1980; and M.S. and Ph.D. degrees in Mechanical Engineering from the University of Iowa in 1987 and 1990, respectively. Since then, he joined the Center for Computer-Aided Design (CCAD) at Iowa as a Research Scientist and shortly after was promoted to CAE Technical Area Manager. In 1997, he joined OU as an Assistant Professor. In 2001, he was promoted to Associative Professor, and in 2005 to the rank of Professor.

Dr. Chang teaches mechanical design and manufacturing, in addition to conducting research in computer-aided modeling and simulation for design and manufacturing of mechanical systems. His work has been published in eight books and more than 150 articles in international journals and conference proceedings. He has also served as a technical consultant to US industry and foreign companies, including LG-Electronics, Seagate Technology, etc. Dr. Chang received numerous awards for his teaching and research in the past few years, including the Williams Companies Foundation presidential professorship in 2005 for meeting the highest standards of excellence in scholarship and teaching, OU Regents Award for Superior Accomplishment in Research and Creative Activity in 2004, OU BP AMOCO Foundation Good Teaching Award in 2002, and OU Regents Award for Superior Teaching in 2010. He is a five-time recipient of CoE Alumni Teaching Award between 2007 and 2009, given to top teachers in CoE. His research paper was given a Best Paper Award at the iCEER-2005 iNEER Conference for Engineering Education and Research in 2005. In 2006, he was awarded a Ralph R. Teetor Educational Award by SAE in recognition of significant contributions to teaching, research, and student development. Dr. Chang was honored by the OKC Mayor's Committee on Disability Concerns with the 2009 Don Davis Award, which is the highest honor granted in public recognition of extraordinarily meritorious service which has substantially advanced opportunities for people with disabilities by removing social, attitudinal and environmental barriers in the greater Oklahoma City area. In 2013, Dr. Chang was named David Ross Boyd Professor, one of the highest honors at the University of Oklahoma, for having consistently demonstrated outstanding teaching, guidance, and leadership for students in an academic discipline or in an interdisciplinary program within the University.

About the Cover

The picture shown on the book cover illustrates the concept of e-Design using a formula SAE (Society of Automotive Engineers) style racecar designed and built by engineering students at the University of Oklahoma (OU). The four pictures on the left show the computer modeling and simulation of the racecar design at numerous stages, including concept design of chassis frame (top right), detailed design of load carrying components (top-left), machining simulation of the wheel center (lower-left), and detailed design of the entire racecar (lower-right). The physical racecar fabricated by the student team in 2006 is shown to the right, which resembles very closely to the computer model (lower-right of the quad pictures). To the author's knowledge, this was the first such detailed CAD model built by engineering students for a racecar. This model was built in Pro/ENGINEER with about 1400 parts and assemblies. The computer model was created in such detail that it was within 0.7 lb. of the as-built car, which weights 445 lb. Among other factors, the e-Design paradigm and computer tools propel the student team from mediocre to a top-ten contender for the annual Formula SAE competitions in only three years.

Acknowledgments

I would like to first thank Mr. Joseph P. Hayton for recognizing the need for such an engineering design book series that offers knowledge in modern engineering design principles, methods, and tools to mechanical engineering students. His enthusiasm in moving the book project forward and eventually publishing the book is highly appreciated. Mr. Hayton's colleagues at Elsevier, Ms. Lisa Jones and her production team, and Ms. Chelsea Johnson, have made significant contributions in transforming the original manuscripts into a well-organized and professionally polished book that is suitable and presentable to our readers.

I am thankful to Dr. Yunxiang Wang, then PhD student of Mechanical Engineering at the University of Oklahoma, for his help in preparing part of the manuscripts. His contribution to this book, especially Chapters 13, 16, 19, and Tutorial Project P1, is highly valuable. Thanks are due to Mr. Matthew Majors, former mechanical engineering student at OU, for his help in preparing tutorial examples for Projects S3 and P3. Thanks are due as well to a student team of former OU students Jimmy Robertson, Chris Erickson, Shashank Ramarao, Matthew Walker, and John Harrison for their excellent work in designing and prototyping the bicycle wind measure device (BWMD), which was employed as a case study for cost estimating in Chapter 15.

I am grateful to my current and former graduate students Dr. Yunxiang Wang, Dr. Mangesh Edke, Dr. Qunli Sun, Dr. Sung-Hwan Joo, Dr. Xiaoming Yu, Dr. Hsiu-Ying Hwang, Mr. Trey Wheeler, Mr. Iulian Grindeanu, Mr. Tyler Bunting, Mr. David Gibson, Mr. Chienchih Chen, Mr. Tim Long, Mr. Poh-Soong Tang, and Mr. Javier Silver, for their excellent efforts in conducting research on numerous aspects of engineering design. Ideas and results that came out of their research have been largely incorporated into this book. Their dedication to the research in developing computer-aided approaches for support of product design modeling and simulation is acknowledged and is highly appreciated.

CHAPTER

INTRODUCTION TO e-DESIGN

1

CHAPTER OUTLINE

1.1 Introduction 2
1.2 The e-Design Paradigm 5
1.3 Virtual Prototyping 7
1.3.1 Parameterized CAD Product Model 7
1.3.1.1 Parameterized Product Model 8
1.3.1.2 Analysis Models 8
1.3.1.3 Motion Simulation Models 10
1.3.2 Product Performance Analysis 11
1.3.2.1 Motion Analysis 11
1.3.2.2 Structural Analysis 11
1.3.2.3 Fatigue and Fracture Analysis 12
1.3.2.4 Product Reliability Evaluations 12
1.3.3 Product Virtual Manufacturing 13
1.3.4 Tool Integration 13
1.3.5 Design Decision Making 15
1.3.5.1 Design Problem Formulation 15
1.3.5.2 Design Sensitivity Analysis 16
1.3.5.3 Parametric Study 16
1.3.5.4 Design Trade-Off Analysis 17
1.3.5.5 What-If Study 19
1.4 Physical Prototyping 19
1.4.1 Rapid Prototyping 19
1.4.2 CNC Machining 21
1.5 Example: Simple Airplane Engine 23
1.5.1 System-Level Design 23
1.5.2 Component-Level Design 25
1.5.3 Design Trade-Off 25
1.5.4 Rapid Prototyping 26
1.6 Example: High-Mobility Multipurpose Wheeled Vehicle 26
1.6.1 Hierarchical Product Model 27
1.6.2 Preliminary Design 28
1.6.3 Detailed Design 30
1.6.4 Design Trade-Off 32

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00001-6
Copyright © 2015 Elsevier Inc. All rights reserved.

1.7 Summary **35**
Questions and Exercises **35**
References **36**

Conventional product development employs a design–build–test philosophy. The sequentially executed development process often results in prolonged lead times and elevated product costs. The proposed e-Design paradigm employs IT-enabled technology for product design, including virtual prototyping (VP) to support a cross-functional team in analyzing product performance, reliability, and manufacturing costs early in product development, and in making quantitative trade-offs for design decision making. Physical prototypes of the product design are then produced using the rapid prototyping (RP) technique and computer numerical control (CNC) to support design verification and functional prototyping, respectively.

e-Design holds potential for shortening the overall product development cycle, improving product quality, and reducing product costs. It offers three concepts and methods for product development bringing product performance, quality, and manufacturing costs together early in design for consideration; supporting design decision making based on quantitative product performance data; incorporating physical prototyping techniques to support design verification and functional prototyping.

1.1 INTRODUCTION

A conventional product development process that is usually conducted sequentially suffers the problem of the *design paradox* (Ullman, 1992). This refers to the dichotomy or mismatch between the design engineer's knowledge about the product and the number of decisions to be made (flexibility) throughout the product development cycle (see Figure 1.1). Major design decisions are usually made in the early design stage when the product is not very well understood. Consequently, engineering changes are frequently requested in later product development stages, when product design evolves and is better understood, to correct decisions made earlier.

Conventional product development is a design–build–test process. Product performance and reliability assessments depend heavily on physical tests, which involve fabricating functional prototypes

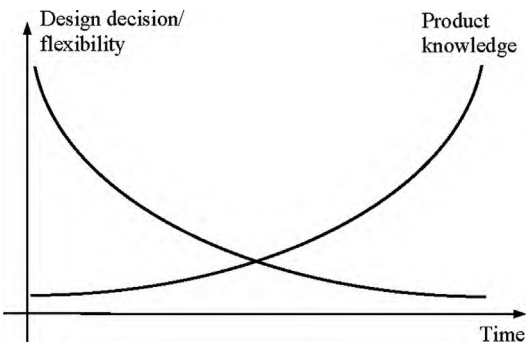


FIGURE 1.1

The design paradox.

of the product and usually lengthy and expensive physical tests. Fabricating prototypes usually involves manufacturing process planning and fixtures and tooling for a very small amount of production. The process can be expensive and lengthy, especially when a design change is requested to correct problems found in physical tests.

In conventional product development, design and manufacturing tend to be disjointed. Often, manufacturability of a product is not considered in design. Manufacturing issues usually appear when the design is finalized and tests are completed. Design defects related to manufacturing in process planning or production are usually found too late to be corrected. Consequently, more manufacturing procedures are necessary for production, resulting in elevated product cost.

With this highly structured and sequential process, the product development cycle tends to be extended, cost is elevated, and product quality is often compromised to avoid further delay. Costs and the number of engineering change requests (ECRs) throughout the product development cycle are often proportional according to the pattern shown in Figure 1.2. It is reported that only 8% of the total product budget is spent on design; however, in the early stage, design determines 80% of the lifetime cost of the product (Anderson, 1990). Realistically, today's industries will not survive worldwide competition unless they introduce new products of better quality, at lower cost, and with shorter lead times. Many approaches and concepts have been proposed over the years, all with a common goal—to shorten the product development cycle, improve product quality, and reduce product cost.

A number of proposed approaches are along the lines of virtual prototyping (Lee, 1999), which is a simulation-based method that helps engineers understand product behavior and make design decisions in a virtual environment. The virtual environment is a computational framework in which the geometric and physical properties of products are accurately simulated and represented. A number of successful virtual prototypes have been reported, such as Boeing's 777 jetliner, General Motors' locomotive engine, Chrysler's automotive interior design, and the Stockholm Metro's Car 2000 (Lee, 1999). In addition to virtual prototyping, the concurrent engineering (CE) concept and methodology have been studied and developed with emphasis on subjects such as product life cycle design, design for X-abilities (DFX), integrated product and process development (IPPD), and Six Sigma (Prasad, 1996).

Although significant research has been conducted in improving the product development process and successful stories have been reported, industry at large is not taking advantage of new product

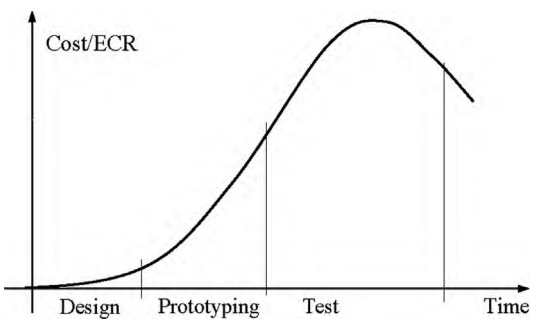


FIGURE 1.2

Cost/ECR versus time in a conventional design cycle.

development paradigms. The main reason is that small and mid-size companies cannot afford to develop an in-house computer tool environment like those of Boeing and the Big-Three automakers. On the other hand, commercial software tools are not tailored to meet the specific needs of individual companies; they often lack proper engineering capabilities to support specific product development needs, and most of them are not properly integrated. Therefore, companies are using commercial tools to support segments of their product development without employing the new design paradigms to their full advantage.

The e-Design paradigm does not supersede any of the approaches discussed. Rather, it is simply a realization of concurrent engineering through virtual and physical prototyping with a systematic and quantitative method for design decision making. Moreover, e-Design specializes in performance and reliability assessment and improvement of complex, large-scale, computer-intensive mechanical systems. The paradigm also uses design for manufacturability (DFM), design for manufacturing and assembly (DFMA), and manufacturing cost estimates through virtual manufacturing process planning and simulation for design considerations.

The objective of this chapter is to present an overview of the e-Design paradigm and the sample tool environment that supports a cross-functional team in simulating and designing mechanical products concurrently in the early design stage. In turn, better-quality products can be designed and manufactured at lower cost. With intensive knowledge of the product gained from simulations, better design decisions can be made, breaking the aforementioned design paradox. With the advancement of computer simulations, more hardware tests can be replaced by computer simulations, thus reducing cost and shortening product development time. The desirable cost and ECR distributions throughout the product development cycle shown in Figure 1.3 can be achieved through the e-Design paradigm.

A typical e-Design software environment can be built using a combination of existing computer-aided design (CAD), computer-aided engineering (CAE), and computer-aided manufacturing (CAM) as the base, and integrating discipline-specific software tools that are commercially available or developed in-house for specific simulation tasks. The main technique in building the e-Design environment is tool integration. Tool integration techniques, including product data models, wrappers, engineering views, and design process management, have been developed (Tsai et al., 1995) and are described in Chapter 17, Design Optimization. This integrated e-Design tool environment allows small and mid-size companies to conduct efficient product development using the e-Design paradigm. The tool environment is flexible so that additional engineering tools can be incorporated with less effort.

In addition, the basis for tool integration, such as product data management (PDM), is well established in commercial CAD tools and so no wheel needs to be reinvented. The e-Design paradigm employs three main concepts and methods for product development:

- Bringing product performance, quality, and manufacturing cost for design considerations in the early design stage through virtual prototyping.
- Supporting design decision making through a quantitative approach for both concept and detailed designs.
- Incorporating product physical prototypes for design verification and functional tests via rapid prototyping and CNC machining, respectively.

In this chapter, the e-Design paradigm is introduced. Then components that make up the paradigm, including knowledge-based engineering (KBE) (Gonzalez and Dankel, 1993), virtual prototyping, and

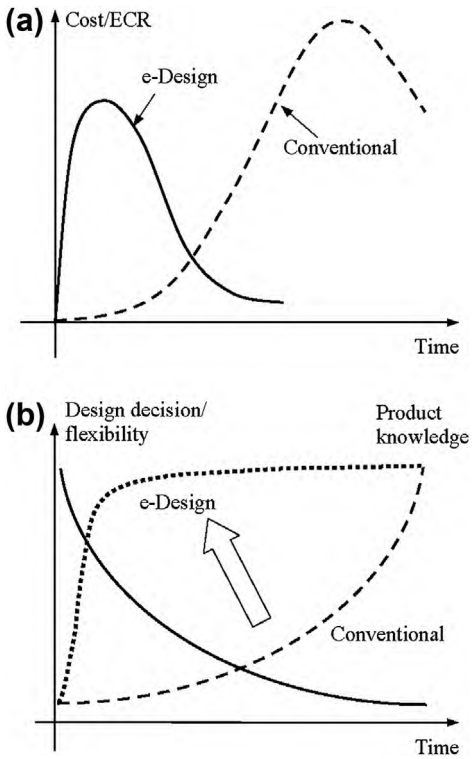


FIGURE 1.3

(a) Cost/ECR versus e-Design cycle time; (b) product knowledge versus e-Design cycle time.

physical prototyping, are briefly presented. Designs of a simple airplane engine and a high-mobility multipurpose wheeled vehicle (HMMWV) are briefly discussed to illustrate the practice of the paradigm. Details of modeling and simulation are provided in later chapters.

1.2 THE e-DESIGN PARADIGM

As shown in Figure 1.4, in e-Design, a product design concept is first realized in a solid model form by design engineers using CAD tools. The initial product is often established based on the designer's experience and legacy data of previous product lines. It is highly desirable to capture and organize designer experience and legacy data to support decision making in a discrete form so as to realize an initial concept. The KBE (Gonzalez and Dankel, 1993) that computerizes knowledge about specific product domains to support design engineers in arriving at a solution to a design problem supports the concept design well. In addition, a KBE system integrated with a CAD tool may directly generate a solid model of the concept design that directly serves downstream design and manufacturing simulations.

With the product solid model represented in CAD, simulations for product performance, reliability, and manufacturing can be conducted. The product development tasks and the cross-functional team are

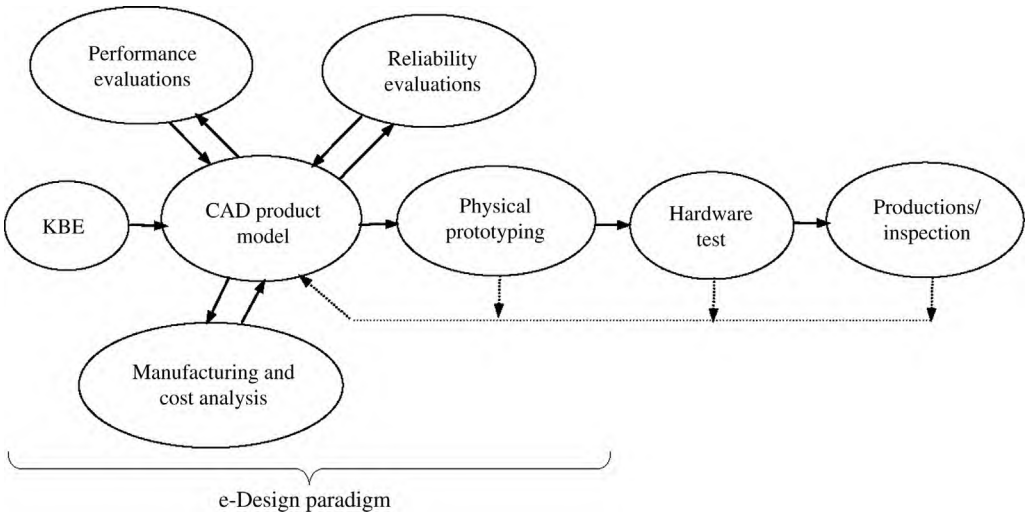


FIGURE 1.4

The e-Design paradigm.

organized according to engineering disciplines and expertise. Based on a centralized computer-aided design product model, simulation models can be derived with proper simplifications and assumptions. However, a one-way mapping that governs changes from CAD models to simulation models must be established for rapid simulation model updates (Chang et al., 1998a). The mapping maintains consistency between CAD and simulation models throughout the product development cycle.

Product performance, reliability, and manufacturing can then be simulated concurrently. Performance, quality, and costs obtained from multidisciplinary simulations are brought together for review by the cross-functional team. Design variables—including geometric dimensions and material properties of the product CAD models that significantly influence performance, quality, and cost—can be identified by the cross-functional team in the CAD product model. These key performance, quality, and cost measures, as well as design variables, constitute a product design model. With such a model, a systematic design approach, including a parametric study for concept design and a trade-off study for detailed design, can be conducted to search for better design alternatives with a minimum number of design iterations.

The product designed in the virtual environment can then be fabricated using rapid prototyping machines for physical prototypes directly from product CAD solid models, without tooling and process planning. The physical prototypes support the cross-functional team for design verification and assembly checking. Change requests that are made at this point can be accommodated in the virtual environment without high cost or delay.

The physics-based simulation technology potentially minimizes the need for product hardware tests. Because substantial modeling and simulations are performed, unexpected design defects encountered during the hardware tests are reduced, thus minimizing the feedback loop for design modifications. Moreover, the production process is smooth since the manufacturing process has been planned and simulated. Potential manufacturing-related problems will have been largely addressed in earlier stages.

A number of commercial CAD systems provide a suite of integrated CAD/CAE/CAM capabilities (e.g., Pro/ENGINEER of Parametric Technology Co., www.ptc.com and SolidWorks®, www.solidworks.com). Other CAD systems, including CATIA® (www.3ds.com/products-services/catia) and NX (www.plm.automation.siemens.com/en_us/products/nx/), support one or more aspects of the engineering analysis. In addition, third-party software companies have made significant efforts in connecting their capabilities to CAD systems. As a representative example, CAE and CAM software companies worked with SolidWorks and integrated their software into SolidWorks environments such as CAMWorks® (www.camworks.com). Each individual tool is seamlessly integrated into SolidWorks.

In this book, Pro/ENGINEER and SolidWorks, with a built-in suite of CAE/CAM modules, are employed as the base for the e-Design environment. In addition to their superior solid modeling capability based on parametric technology (Zeid, 1991), Pro/MECHANICA® and SolidWorks Simulation support simulations of basic engineering problems, including structural and thermal. Mechanism Design of Pro/ENGINEER and SolidWorks Motion support motion simulation of mechanical systems. Moreover, CAM capabilities implemented in CAD, such as Pro/MFG, and CAMWorks, provide an excellent basis for manufacturing process planning and simulations. Additional CAD/CAE/CAM tools introduced to support modeling and simulation of broader engineering problems encountered in general mechanical systems can be developed and added to the tool environment as needed.

1.3 VIRTUAL PROTOTYPING

Virtual prototyping is the backbone of the e-Design paradigm. As presented in this chapter, VP consists of constructing a parametric product model in CAD, conducting product performance simulations and reliability evaluations using CAE software, and carrying out manufacturing simulations and cost estimating using CAM software. Product modeling and simulations using integrated CAD/CAE/CAM software are the basic and common activities involved in virtual prototyping. However, a systematic design method, including parametric study and design trade-offs, is indispensable for design decision making.

1.3.1 PARAMETERIZED CAD PRODUCT MODEL

A parametric product model in CAD is essential to the e-Design paradigm. The product model evolves to a higher-fidelity level from concept to detailed design stages (Chang et al., 1998a). In the concept design stage, a considerable portion of the product may contain non-CAD data. For example, when the gross motion of the mechanical system is sought, the non-CAD data may include engine, tires, or transmission if a ground vehicle is being designed. Engineering characteristics of the non-CAD parts and assemblies are usually described by engineering parameters, physics laws, or mathematical equations. This non-CAD representation is often added to the product model in the concept design stage for a complete product model. As the design evolves, non-CAD parts and assemblies are refined into solid-model forms for subsystem and component designs as well as for manufacturing process planning.

A primary challenge in conducting product performance simulations is generating simulation models and maintaining consistency between CAD and simulation models through mapping.

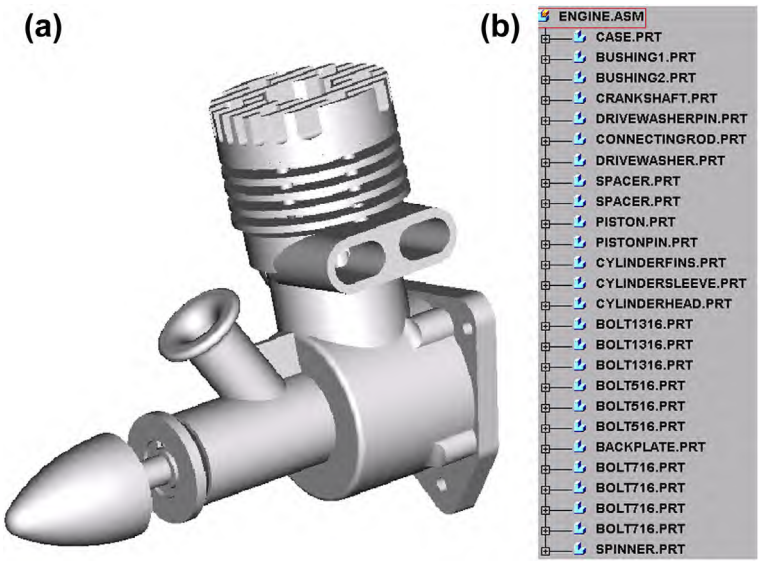


FIGURE 1.5

Airplane engine model: (a) CAD model and (b) model tree.

Challenges involved in model generation and in structural and dynamic simulations are discussed next, in which an airplane engine model in the detail design stage, as shown in Figure 1.5, is used for illustration.

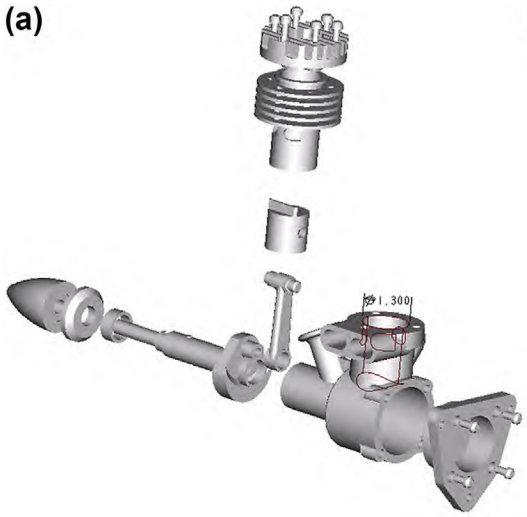
1.3.1.1 Parameterized Product Model

A parameterized product model defined in CAD allows design engineers to conveniently explore design alternatives for support of product design. The CAD product model is parameterized by defining dimensions that govern the geometry of parts through geometric features and by establishing relations between dimensions within and across parts. Through dimensions and relations, changes can be made simply by modifying a few dimensional values. Changes are propagated automatically throughout the CAD product model following the dimensions and relations. A single-piston airplane engine with a change in its bore diameter is shown in Figure 1.6, which illustrates change propagation through parametric dimensions and relationships. More in-depth discussion of the modeling and parameterization of the engine example can be found in Chapter 5: Design Parameterization.

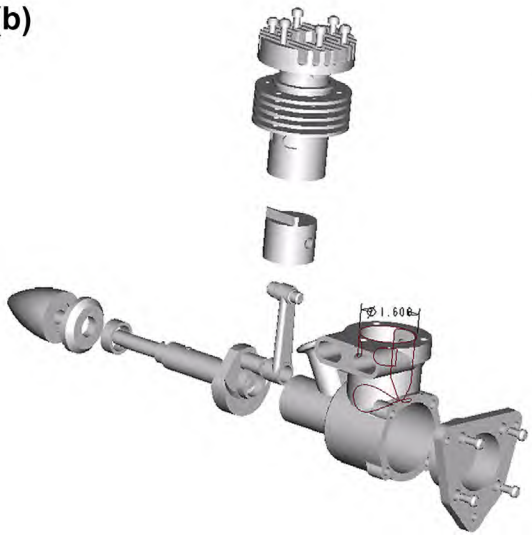
1.3.1.2 Analysis Models

For product structural analysis, finite element analysis (FEA) is often employed. In addition to structural geometry; loads, boundary conditions, and material properties can be conveniently defined in the CAD model. Most CAD tools are equipped with fully automatic mesh generation capability. This capability is convenient but often leads to large FEA models with some geometric discrepancy at the part boundary. Plus, triangular and tetrahedral elements are often the only

(a)



(b)



(c)

Relation	Parameter	New value
/*** Relations for ENGINE:		
/* CASE		
D55:0=D46:0	D55:0	1.416000e+00
D43:0=D46:0/2+0.424	D43:0	1.132000e+00
D40:0=D46:0/2+0.208667	D40:0	9.166670e-01
D0:0=D40:0-0.002	D0:0	9.146670e-01
D1:0=D40:0+0.124667	D1:0	1.041334e+00
D22:0=D1:0+D0:0-0.208	D22:0	1.748001e+00
D45:0=D22:0	D45:0	1.748001e+00
D47:0=D46:0/2+0.833	D47:0	1.541000e+00
D65:0=D46:0/2-0.332	D65:0	3.760000e-01
D66:0=D46:0/2+0.02133	D66:0	7.293300e-01
D113:0=D46:0/2	D113:0	7.080000e-01
D85:0=D0:0+0.40188	D85:0	1.316547e+00
D156:0=D0:0+D1:0	D156:0	1.956001e+00
D203:0=D0:0+D1:0	D203:0	1.956001e+00
D132:0=D46:0/2	D132:0	7.080000e-01
D282:0=(D46:0/2+D43:0)/2	D282:0	9.200000e-01
/* CRANKSHAFT		
D12:6=(D46:0/2-0.44444)*2	D12:6	5.271200e-01
/* CONNECTING ROD		
D27:10=D46:0/2-.23333	D27:10	4.746700e-01
/* CYLINDER FINS		
D42:26=D282:0	D42:26	9.200000e-01
D45:26=D282:0	D45:26	9.200000e-01
D0:26=D46:0/2	D0:26	7.080000e-01
D1:26=D0:26+0.122	D1:26	8.300000e-01
D2:26=D43:0+0.118	D2:26	1.250000e+00
/* PISTON		
D1:16=D46:0/2-0.0827	D1:16	6.253000e-01
D7:16=D1:16-0.2413	D7:16	3.840000e-01
D22:16=D1:16-0.0653	D22:16	5.600000e-01
D19:16=D22:16*2-0.287	D19:16	8.330000e-01
/* CYLINDER SLEEVE		
D5:28=D46:0/2	D5:28	7.080000e-01
D7:28=D46:0-0.165	D7:28	1.251000e+00
D3:28=D46:0/2+0.08333	D3:28	7.913300e-01
/* CYLINDER HEAD		
D141:30=D282:0	D141:30	9.200000e-01
D123:30=D7:16-D8:16/2	D123:30	3.420000e-01
D0:30=D2:26	D0:30	1.250000e+00
D1:30=D1:16	D1:30	6.253000e-01

FIGURE 1.6

Design change propagation: (a) bore diameter = 1.3 in.; (b) bore diameter changed to 1.6 in.; (c) relations of geometric dimensions.

elements supported. An engine connecting rod example meshed using Pro/MESH (part of Pro/MECHANICA) with default mesh parameters is shown in Figure 1.7. The FEA model consists of 1,270 nodes and 4,800 tetrahedron elements, yet it still reveals discrepancy to the true CAD geometry. Moreover, mesh distortion due to large deformation of the structure, such as hyperelastic

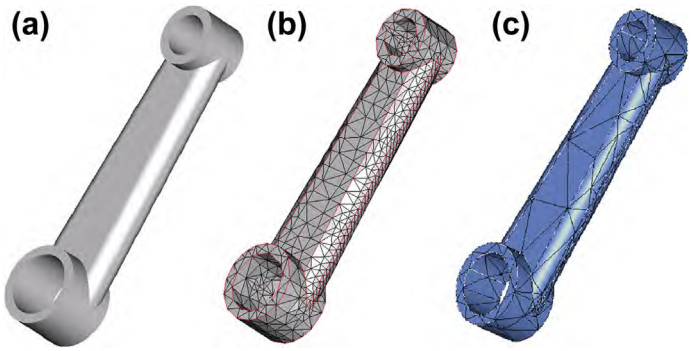


FIGURE 1.7

Finite element meshes of a connecting rod: (a) CAD solid model, (b) h-version finite element mesh, and (c) p-version finite element mesh.

problems, often causes FEA to abort prematurely. Semiautomatic mesh generation is more realistic; therefore, tools such as MSC/Patran® (MacNeal-Schwendler Corp., www.mscsoftware.com) and HyperMesh® (Altair® Engineering, Inc., www.altair.com) are essential to support the e-Design environment for mesh generation.

In general, p-version FEA (Szabó and Babuška, 1991) is more suitable for structural analysis in terms of minimizing the gap in geometry between CAD and finite element models, and in lessening the tendency toward mesh distortion. It also offers capability in convergence analysis that is superior to regular h-version FEA. As shown in Figure 1.7(c), the same connecting rod is meshed with 568 tetrahedron p-elements, using Pro/MECHANICA with a default setting. A one-way mapping between changes in CAD geometric dimensions and finite element mesh for both h- and p-version FEAs can be established through a design velocity field (Haug et al., 1986), which allows direct and automatic generation of the finite element mesh of new designs.

Another issue worth considering is the simplification of 3D solid models to surface (shell) or curve (beam) models for analysis. Capabilities that semiautomatically convert 3D thin-shell solids to surface models are available in, for example, Pro/MECHANICA and SolidWorks Simulation.

1.3.1.3 Motion Simulation Models

Generating motion simulation models involves regrouping parts and subassemblies of the mechanical system in CAD as bodies and often introducing non-CAD components to support a multibody dynamic simulation (Haug, 1989). Engineers must define the joints or force connections between bodies, including joint type and reference coordinates. Mass properties of each body are computed by CAD with the material properties specified. Integration between Mechanism Design and Pro/ENGINEER, as well as between SolidWorks Motion and SolidWorks, is seamless. Design changes made in geometric dimensions propagate to the motion model directly. In addition, simulation tools, such as Dynamic Analysis and Design Systems (DADS) (LMS, www.lmsintl.com/DADS) are integrated with CAD with proper parametric mapping that support parametric study. As an example, the motion inside an airplane engine is modeled as a slider-crank mechanism in Mechanism Design, as shown in Figure 1.8.

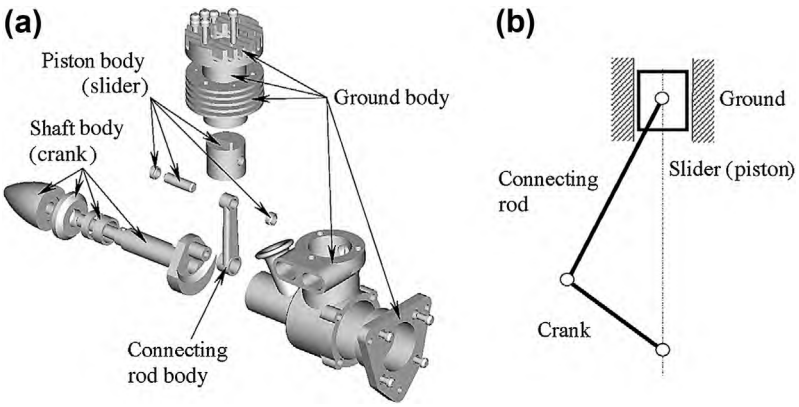


FIGURE 1.8

Engine motion model: (a) definition and (b) schematic view.

A common mistake made in creating motion simulation models is selecting improper joints to connect bodies. Introducing improper joints creates an invalid or inaccurate model that does not simulate the true behavior of the mechanical system. Intelligent modeling capability that automatically specifies joints in accordance with assembly relations defined between parts and subassemblies in solid models is available in, for example, SolidWorks Motion.

1.3.2 PRODUCT PERFORMANCE ANALYSIS

As mentioned earlier, product performance evaluation using physics-based simulation in the computer environment is usually called, in a narrow sense, virtual prototyping, or VP. With the advancement of simulation technology, more engineering questions can be answered realistically through simulations, thus minimizing the need for physical tests. However, some key questions cannot be answered for sophisticated engineering problems—for example, the crashworthiness of ground vehicles. Although VP will probably never replace hardware tests completely, the savings it achieves for less sophisticated problems is significant and beneficial.

1.3.2.1 Motion Analysis

System motion simulations include workspace analysis (kinematics), rigid- and flexible-body dynamics, and inverse dynamic analysis. Mechanism Design and SolidWorks Motion, based on theoretical work of Kane and Levinson (1985), mainly support kinematics and rigid-body simulations for mechanical systems. They do not properly support mechanical system simulation such as a vehicle moving on a user-defined terrain. General-purpose dynamic simulation tools, such as DADS or Adams® (www.mscsoftware.com), are more desirable for simulation of general mechanical systems.

1.3.2.2 Structural Analysis

Pro/MECHANICA supports linear static, vibration, buckling, fatigue, and other such analyses, using p-version FEA. General-purpose finite element codes, such as MSC/Nastran® (MacNeal-Schwendler Corp., www.mscsoftware.com) and ANSYS® (ANSYS Analysis Systems, Inc., www.ansys.com) are

ideal for the e-Design environment to support FEA for a broad range of structural problems—for example, nonlinear, plasticity, and transient dynamics. Meshless methods developed in recent years (for example, Chen et al., 1997) hold promise for avoiding finite element mesh distortion in large-deformation problems. Multiphase problems (e.g., acoustic and aero-structural) are well supported by specialized tools such as LMS® SYSNOISE (Numerical Integration Technologies, 1998, www.lmsintl.com/SYSNOISE). LS-DYNA® (Hallquist, 2006, www.lstc.com) is currently one of the best codes for nonlinear, plastic, dynamic, friction-contact, and crashworthiness problems. These special codes provide excellent engineering analysis capabilities that complement those provided in CAD systems.

1.3.2.3 Fatigue and Fracture Analysis

Fatigue and fracture problems are commonly encountered in mechanical components because of repeated mechanical or thermal loads. MSC Fatigue® (MacNeal-Schwendler Corp., www.mscsoftware.com), with an underlying computational engine developed by nCode® (www.ncode.com) is one of the leading fatigue and fracture analysis tools. It offers both high- and low-cycle fatigue analyses. A critical plane approach is available in MSC Fatigue, www.mscsoftware.com for the prediction of fatigue life due to general multiaxial loads.

Note that the recently developed extended finite element method (XFEM) supports fracture propagation without re-meshing (Moës et al., 2002). XFEM was recently integrated in ABAQUS® (www.3ds.com/products-services/simulia/products/abaqus). Also note that additional capabilities, such as thermal analysis, computational fluid dynamics (CFD) and combustion, can be added to meet specific needs in analyzing mechanical products. Integration of additional engineering disciplines is briefly discussed in Section 1.3.4.

1.3.2.4 Product Reliability Evaluations

Product reliability evaluations in the e-Design environment focus on the probability of specific failure events (or failure mode). The failure event corresponds to a product performance measure, such as the fatigue life of a mechanical component. For the reliability analysis of a single failure event, the failure event or failure function is defined as (Madsen et al., 1986)

$$g(\boldsymbol{X}) = \psi^u - \psi(\boldsymbol{X}) \tag{1.1}$$

where

ψ is a product performance measure
ψ^u is the upper bound (or design requirement) of the product performance
$\boldsymbol{X}$ is a vector of random variables.

When product performance does not meet the requirement—that is, when $\psi^u \leq \psi(\boldsymbol{X})$, the event fails. Therefore, the probability of failure P_f of the particular event $g(\boldsymbol{X}) \leq 0$ is

$$P_f = P[g(X) \leq 0] \tag{1.2}$$

where $P[\bullet]$ is the probability of event $\bullet$.

Given the joint probability density function $f_{\boldsymbol{X}}(\boldsymbol{x})$ of the random variables $\boldsymbol{X}$, the probability of failure for a single event of a mechanical component can be expressed as

$$P_f = P[g(\boldsymbol{X}) \leq 0] = \underset{g(\boldsymbol{X})\leq 0}{\int\int} \ldots \int f_{\boldsymbol{X}}(\boldsymbol{x}) d\boldsymbol{x}. \tag{1.3}$$

The probability of failure in Eq. 1.3 is commonly evaluated using the Monte Carlo method or the first- or second-order reliability method (FORM or SORM) (Wu and Wirsching, 1984; Yu et al., 1998).

Once the probabilities of several failure events in subsystems or components are computed, system reliability can be obtained by using, for example, fault-tree analysis (Ertas and Jones, 1993). No general-purpose software tool for reliability analysis of general mechanical systems is commercially available yet. Numerical evaluation of stochastic structures under stress (NESSUS®) (www.nessus.swri.org), which is currently in development can be a good candidate for incorporation into the e-Design environment. With the probability of failure, critical quality design criteria, such as mean time between failure (MTBF), can be computed (Ertas and Jones, 1993).

Two main challenges exist in reliability analysis: One, realistic distribution data are difficult to acquire and often are not available in the early stage; and two, failure probability computations are often expensive. The first challenge may be alleviated by employing legacy data from previous product lines. Approximation techniques (e.g., Yu et al., 1998) can be employed to make the computation affordable for individual failure events within a mechanical component.

1.3.3 PRODUCT VIRTUAL MANUFACTURING

Virtual manufacturing addresses issues of design for manufacturability (DFM) (Prasad, 1996) and design for manufacturing and assembly (DFMA) (Boothroyd et al., 1994) early in product development. In the e-Design paradigm, DFM and DFMA are performed by conducting virtual manufacturing and assembly using, for example, Pro/MFG. DFM and DFMA of the product are verified through animations of the virtual manufacturing and assembly process.

Pro/MFG is a Pro/ENGINEER module supporting the virtual machining process, including milling, drilling, and turning. By incorporating part design and also defining workpieces, workcells, fixtures, cutting tools, and cutting parameters, Pro/MFG automatically generates a toolpath (see Figure 1.9(a)), simulates the machining process (Figure 1.9(b)), calculates machining time, and produces cutter location (CL) data. The CL data can be post-processed for CNC codes. In addition, casting, sheet metal, molding, and welding can be simulated using Pro/CASTING, Pro/SHEETMETAL, Pro/MOLD, and Pro/WELDING, respectively.

With such virtual manufacturing process planning and animation, manufacturability of the product design can, to some extent, be verified. The DFMA tool (Boothroyd et al., 1994) developed by Boothroyd Dewhurst, Inc., assists the cross-functional team in quantifying product assembly time and labor costs. It also challenges the team to simplify product structure, thereby reducing product as well as assembly costs.

One of the limitations in using virtual manufacturing tools (e.g., Pro/MFG) is that chip formation (Fang and Jawahir, 1996), a primary consideration in computer numerical control (CNC), is not incorporated into the simulation. In addition, machining parameters, such as power consumption, machining temperature, and tool life, which contribute to manufacturing costs are not yet simulated.

1.3.4 TOOL INTEGRATION

Techniques developed to support tool integration (Chang et al., 1998a) include parameterized product data models, engineering views, tool wrappers, and design process management. Parameterized

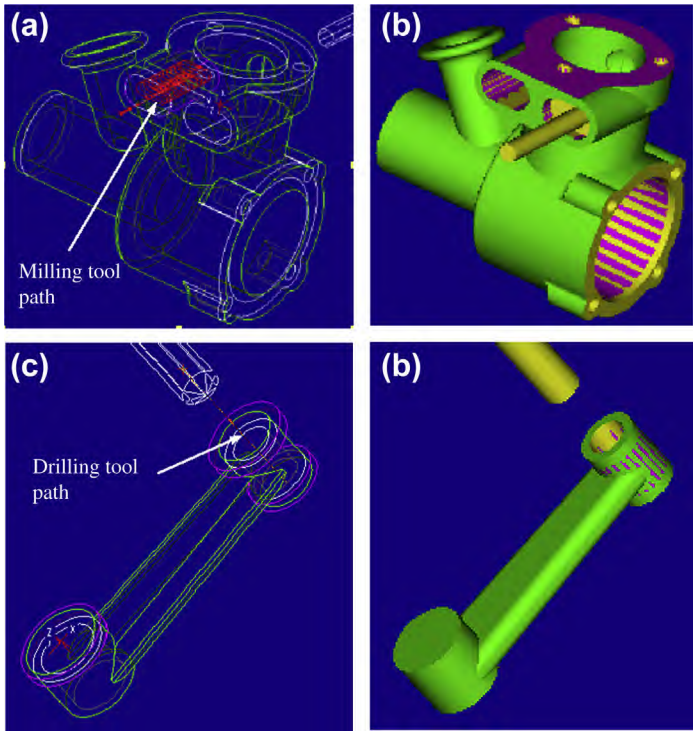


FIGURE 1.9

Virtual machining process: (a) engine case—milling toolpath; (b) milling simulation; (c) connecting rod—drilling toolpath; (d) drilling simulation.

product data models represent engineering data that are needed for conducting virtual prototyping of the mechanical system. The main sources of the product data model are CAD and non-CAD models. The product data model evolves throughout the product development cycle as illustrated in Figure 1.10.

Engineering views allow engineers from various disciplines to view the product from their own technical perspectives. Through engineering views, engineers create simulation models that are consistent with the product model by simplifying the CAD representation as needed and adding non-CAD product representation and mapping. Tool wrappers provide two-way data translation and transmission between engineering tools and the product data model. Design process management provides the team leader with a tool to monitor and manage the design process. When a new tool of an existing discipline, for example ANSYS for structural FEA, is to be integrated, a wrapper for it must be developed. Three main tasks must be carried out when a new engineering discipline, say computational fluid dynamics (CFD), is added to the environment. First, the product data model must be extended to include engineering data needed to support CFD. Second, engineering views must be added to allow design engineers to generate CFD models. Finally, wrappers must be developed for specific CFD tools.

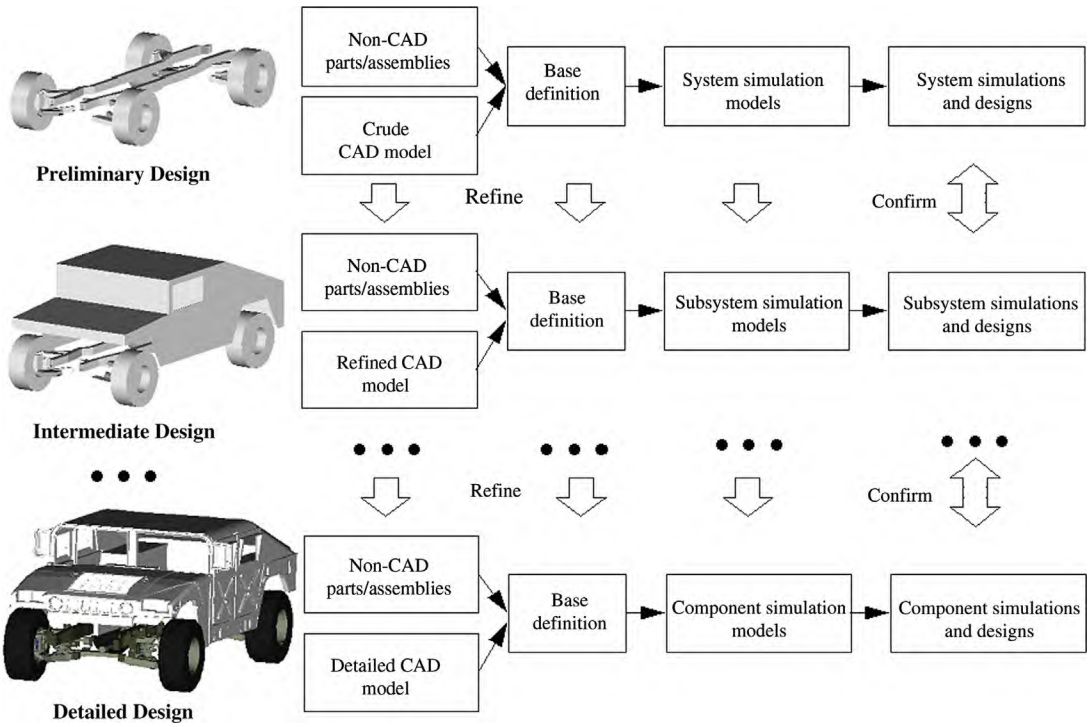


FIGURE 1.10

Hierarchical product models evolved through the e-Design process.

1.3.5 DESIGN DECISION MAKING

Product performance, reliability, and manufacturing cost that are evaluated using simulations can be brought to the cross-functional team for review. Product performance and reliability are checked against product specifications that have been defined and have evolved from the beginning of the product development process. Manufacturing cost derived from the virtual manufacturing simulations can be added to product cost. The cross-functional team must address areas of concern identified in product performance, reliability, and manufacturability, and it must identify a set of design variables that influence these areas. Design modifications can then be conducted. In the past, quality functional deployment (QFD) (Ertas and Jones, 1993) was largely employed in design modification to assign qualitative weighting factors to product performance and design changes. e-Design employs a systematic and quantitative approach to design modifications (for example, Yu et al., 1997).

1.3.5.1 Design Problem Formulation

Before a design can be improved, design problems must be defined. A design problem is often presented in a mathematical form, typically as

$$\text{Minimize } \phi(\boldsymbol{b}) \tag{1.4a}$$

Subject to

$$\psi_i(\boldsymbol{b}) \le \psi_i^u \quad i = 1, m \tag{1.4b}$$

$$P_{f_j}(\boldsymbol{b}) \le P_{f_j}^u \quad j = 1, n \tag{1.4c}$$

$$b_k^\ell \le b_k \le b_k^u \quad k = 1, p \tag{1.4d}$$

where

$\phi(\boldsymbol{b})$ is the objective (or cost) function to be minimized
$\psi_i(\boldsymbol{b})$ is the ith constraint function that must be no greater than its upper bound ψ_i^u
$P_{f_j}(\boldsymbol{b})$ is the jth failure probability index that must be no greater than its upper bound $P_{f_j}^u$
$\boldsymbol{b}$ is the vector of design variables
b_k^ℓ and b_k^u are the lower and upper bounds of the design variable b_k, respectively.

Note that in e-Design design variables are usually associated with dimensions of geometric features and part material properties in the parameterized CAD models. The feature-based design variables serve as the common language to support the cross-functional team while conducting parametric study and design trade-offs.

1.3.5.2 Design Sensitivity Analysis

Before quantitative design decisions can be made, there must be a design sensitivity analysis (DSA) that computes derivatives of performance measures, including product performance, failure probability, and manufacturing cost, with respect to design variables. Dependence of performance measures on design variables is usually implicit. How to express product performance in terms of design variables in a mathematical form is not straightforward. Analytical DSA methods combined with numerical computations have been developed mainly for structural responses (Haug et al., 1986) and fatigue and fracture (Chang et al., 1997). DSA for failure probability with respect to both deterministic and random variables has also been developed (Yu et al., 1997). In addition, DSA and optimization using meshless methods have been developed for large-deformation problems (Grindeanu et al., 1999). For more details about the analytical DSA for structural responses also refer to Haug et al. (1986).

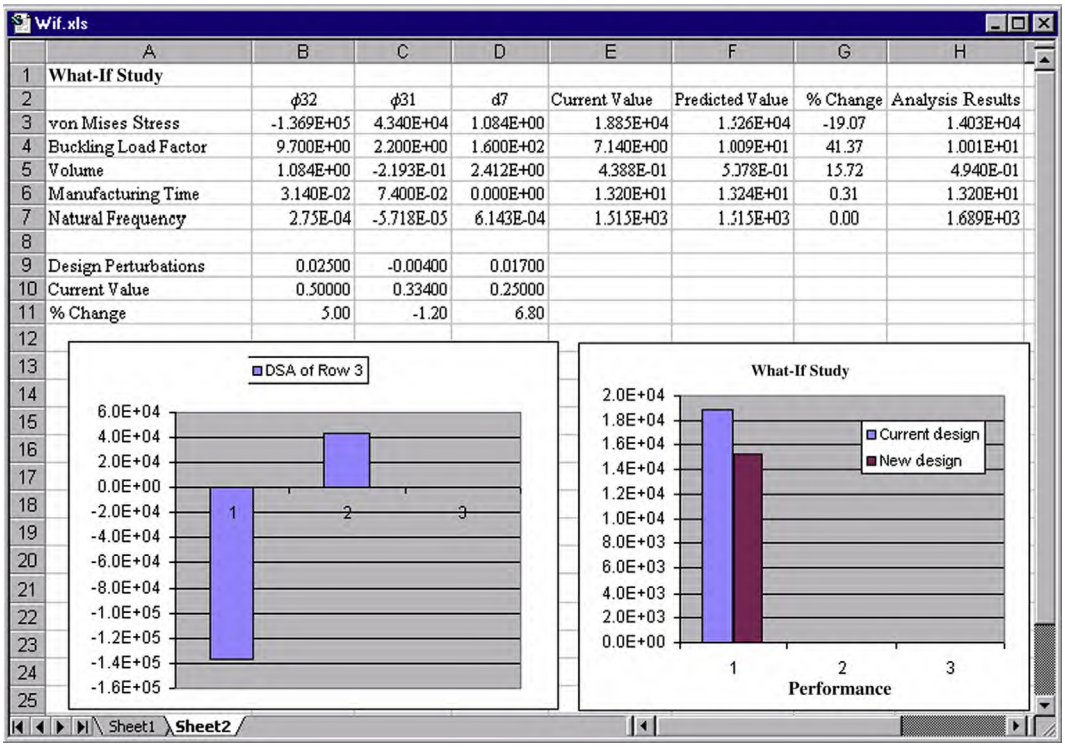
For problems such as motion and manufacturing cost, where premature or no analytical DSA capability is available, the finite difference method is the only choice. The finite difference method is expressed in the following equation:

$$\frac{\partial \psi}{\partial b_j} \approx \frac{\psi(\boldsymbol{b} + \Delta b_j) - \psi(\boldsymbol{b})}{\Delta b_j} \tag{1.5}$$

where Δb_j is a perturbation in the jth design variable. With sensitivity information, parametric study and design trade-offs can be conducted for design improvements at the concept and detail stages, respectively.

1.3.5.3 Parametric Study

A parametric study that perturbs design variables in the product design model to explore design alternatives can effectively support product concept designs. A parametric study is simple and easy to perform as long as the mapping between CAD and simulation models has been established. The



	A	B	C	D	E	F	G	H
1	What-If Study							
2		ϕ32	ϕ31	d7	Current Value	Predicted Value	% Change	Analysis Results
3	von Mises Stress	-1.369E+05	4.340E+04	1.084E+00	1.885E+04	1.526E+04	-19.07	1.403E+04
4	Buckling Load Factor	9.700E+00	2.200E+00	1.600E+02	7.140E+00	1.009E+01	41.37	1.001E+01
5	Volume	1.084E+00	-2.193E-01	2.412E+00	4.388E-01	5.078E-01	15.72	4.940E-01
6	Manufacturing Time	3.140E-02	7.400E-02	0.000E+00	1.320E+01	1.324E+01	0.31	1.320E+01
7	Natural Frequency	2.75E-04	-5.718E-05	6.143E-04	1.515E+03	1.515E+03	0.00	1.689E+03
8								
9	Design Perturbations	0.02500	-0.00400	0.01700				
10	Current Value	0.50000	0.33400	0.25000				
11	% Change	5.00	-1.20	6.80				

FIGURE 1.11

Spreadsheet for parametric study and design trade-offs.

mapping supports fast simulation model generation for performance analyses. It also supports DSA using the finite difference method. The parametric study is possible for concept design because the number of design variables to perturb is usually small. A spreadsheet with a proper formula defined among cells is well suited to support the parametric study. The use of Microsoft Excel is illustrated in Figure 1.11.

1.3.5.4 Design Trade-Off Analysis

With design trade-off analysis, the design engineer can find the most appropriate design search direction for the design problem formulated in Eq. 1.4, using four possible algorithms:

- Reduce cost.
- Correct constraint neglecting cost.
- Correct constraint with a constant cost.
- Correct constraint with a cost increment.

As a general rule, the first algorithm, reduce cost, can be chosen when the design is feasible; in other words, all constraint functions are within the desired limits. When the design is infeasible, generally one may start with the third algorithm, correct constraint with a constant cost. If the design

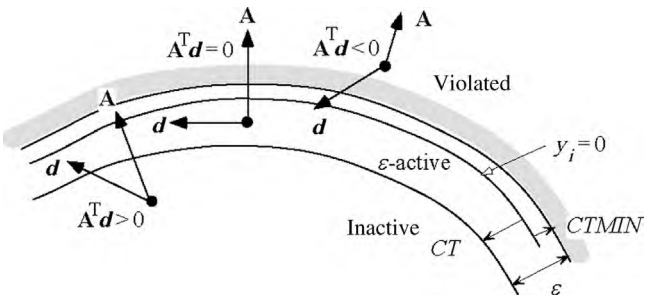


FIGURE 1.12

ε-Active constraint strategy.

remains infeasible, the fourth algorithm, correct constraint with a cost increment—say 10%—may be appropriate. If a feasible design is still not found, the second algorithm, correct constraint neglecting cost, can be selected. A quadratic programming (QP) subproblem can be formulated to numerically find the search direction that corresponds to the algorithm selected.

An ε-active constraint strategy (Arora, 1989), shown in Figure 1.12, can be employed to support design trade-offs. The constraint functions in Eq. 1.4 are normalized by

$$y_i = \frac{\psi_i}{\psi_i^u} - 1 \leq 0, \quad i = 1, m \tag{1.6}$$

when y_i is between *CT* (usually 0.03) and *CTMIN* (usually 0.005), it is active—that is, $\varepsilon = |CT| + CTMIN$, as illustrated in Figure 1.12. When y_i is less than *CT*, the constraint function is inactive or feasible. When y_i is larger than *CTMIN*, the constraint function is violated. A QP subproblem can be formulated to find the search direction numerically corresponding to the option selected. For example, the QP subproblem for the first algorithm (cost reduction) can be formulated as

$$\begin{aligned} &\text{Minimize} && \boldsymbol{c}^T\boldsymbol{d} + 0.5\ \boldsymbol{d}^T\boldsymbol{d} \\ &\text{Subject to} && \mathbf{A}^T\boldsymbol{d} \leq \boldsymbol{y} \\ & && \boldsymbol{b}^L - \boldsymbol{b}^{(k)} \leq \boldsymbol{d} \leq \boldsymbol{b}^U - \boldsymbol{b}^{(k)} \end{aligned} \tag{1.7}$$

where

$$\boldsymbol{c} = [c_1, c_2, \ldots, c_{n1+n2}]^T, \quad c_i = \partial\phi/\partial b_i$$

$\boldsymbol{d}$ is the search direction to be determined.

$$\mathbf{A}_{ij} = \partial P_{y_i} \Big/ \partial b_j; \quad \boldsymbol{b} = [b_1, b_2, \ldots b_n]^T$$

k is the current design iteration.

The objective of the design trade-off algorithm is to find the optimal search direction $\boldsymbol{d}$ under a given circumstance. Details are discussed in Chapter 17 Design Optimization.

1.3.5.5 What-If Study

After the search direction $\boldsymbol{d}$ is found, a number of step sizes α can be used to perturb the design along the search direction $\boldsymbol{d}$. Objective and constraint function values, represented as ψ_i, at a perturbed design $\boldsymbol{b} + \alpha\boldsymbol{d}$ can be approximated using the first-order sensitivity information of the functions by Taylor series expansion about the current design $\boldsymbol{b}$ without going through simulations; that is,

$$\psi_i(\boldsymbol{b} + \alpha\boldsymbol{d}) \approx \psi_i(\boldsymbol{b}) + \frac{\partial \psi_i}{\partial \boldsymbol{b}} \alpha\boldsymbol{d}. \tag{1.8}$$

Note that since there is no analysis involved, the what-if study can be carried out very efficiently. This allows the design engineer to explore design alternatives more effectively.

Once a satisfactory design is identified, after trying out different step sizes α in an approximate sense, the design model can be updated to the new design and then simulations of the new design can be carried out. Eq. 1.8 also supports parametric study, in which the design perturbation $\delta\boldsymbol{b}$ is determined by engineers based on sensitivity information. To ensure a reasonably accurate function prediction using Eq. 1.8, the step sizes must be small so that the perturbation $\partial\psi_i/(\partial\boldsymbol{b})(\alpha\boldsymbol{d})$ is, as a rule of thumb, less than 10% of the function value $\psi_i(\boldsymbol{b})$.

1.4 PHYSICAL PROTOTYPING

In general, two techniques are suitable for fabricating physical prototypes of the product in the design process: rapid prototyping (RP) and computer numerical control (CNC) machining. RP systems, based on solid freeform fabrication (SFF) technology (Jacobs, 1994), fabricate physical prototypes of the product for design verification. The CNC machining fabricates functional parts as well as the mold or die for mass production of the product.

1.4.1 RAPID PROTOTYPING

The Solid Freeform Fabrication (SFF) technology, also called Rapid Prototyping (RP), is an additive process that employs a layer-building technique based on horizontal cross-sectional data from a 3D CAD model. Beginning with the bottom-most cross-section of the CAD model, the rapid prototyping machine creates a thin layer of material by slicing the model into so-called 2½ D layers. The system then creates an additional layer on top of the first based on the next higher cross-section. The process repeats until the part is completely built. It is illustrated using an engine case in the example shown in Figure 1.13. Rapid prototyping systems are capable of creating parts with small internal cavities and complex geometry.

Most important, SFF follows the same layering process for any given 3D CAD models, so it requires neither tooling nor manufacturing process planning for prototyping, as required by conventional manufacturing methods. Based on CAD solid models, the SFF technique fabricates physical prototypes of the product in a short turnaround time for design verification. It also supports tooling for product manufacturing, such as mold or die fabrications, through, for example, investment casting (Kalpakjian, 1992).

Note that there are various types of SFF systems commercially available, such as the SLA® 7000 and Sinterstation® by 3D Systems (Figures 1.14(a) and 1.14(b), www.3dsystems.com). In this chapter,

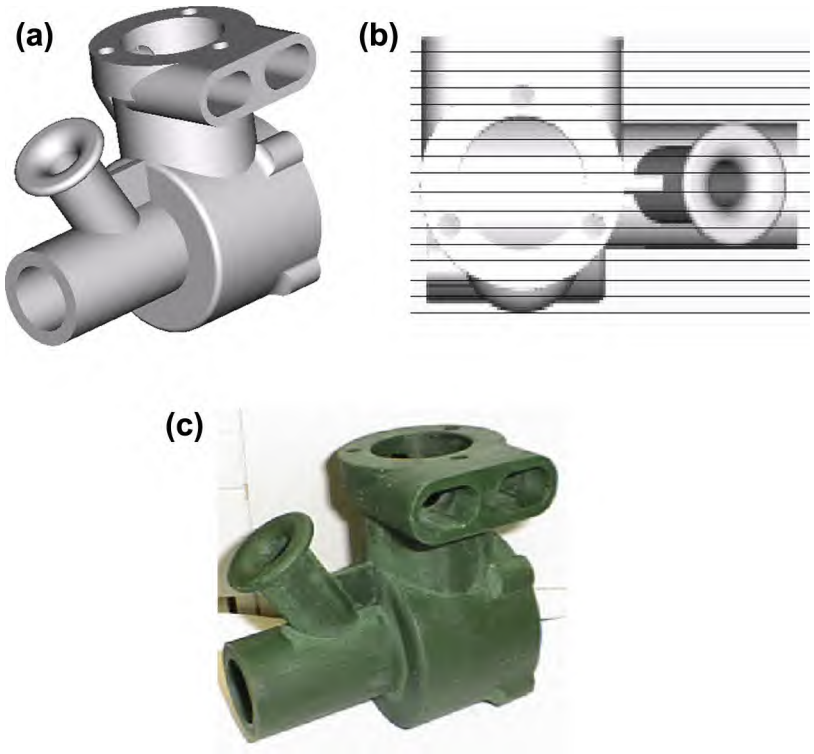


FIGURE 1.13

SFF: layered manufacturing: (a) 3D CAD model, (b) 2-1/2D slicing, and (c) physical model.

the Dimension 1200 sst® machine (www.stratasys.com), as shown in Figure 1.14(c), is presented. More details about it as well as other RP systems will be discussed in Chapter 14 Rapid Prototyping.

The CAD solid model of the product is first converted into a stereolithographic (STL) format (Chua and Leong, 1998), which is a faceted boundary representation uniformly accepted by the industry. Both the coarse and refined STL models of an engine case are shown in Figure 1.15. Even though the STL model is an approximation of the true CAD geometry, increasing the number of triangles can minimize the geometric error effectively. This can be achieved by specifying a smaller chord length, which is defined as the maximum distance between the true geometric boundary and the neighboring edge of the triangle. The faceted representation is then sliced into a series of 2D sections along a prespecified direction. The slicing software is SFF-system dependent.

The Dimension 1200 sst employs fused deposition manufacturing (FDM) technology. Acrylonitrile butadiene styrene (ABS) materials are softened (by elevating temperature), squeezed through a nozzle on the print heads, and laid on the substrate as build and support materials, respectively, following the 2D contours sliced from the 3D solid model (Figure 1.16). Note that various crosshatch options are available in CatalystEX® software (www.dimensionprinting.com), which comes with the rapid prototyping system.

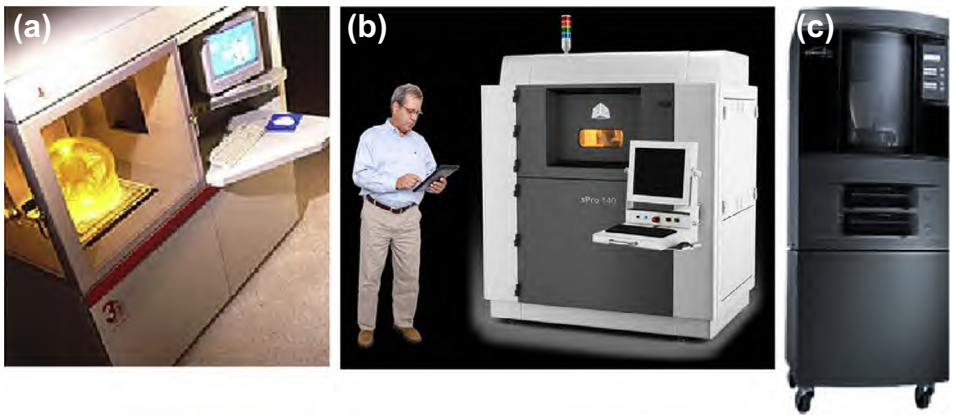


FIGURE 1.14

Commercial RP systems: (a) 3D systems' SLA 7000, (b) Sinterstation 2500, (c) Stratasys inc.'s dimension 1200 sst. (Sources: (b) 3D Systems Corporation, USA; (c) Stratasys Ltd.)

The physical prototypes are mainly for the cross-functional team to verify the product design and check the assembly. However, they can also be used for discussion with marketing personnel to develop marketing ideas. In addition, the prototypes can be given to potential customers for feedback, thus bringing customers into the design loop early in product development.

1.4.2 CNC MACHINING

The machining operations of virtual manufacturing, such as milling, turning, and drilling, allow designers to plan the machining process, generate the machining toolpath, visualize and simulate machining operations, and estimate machining time. Moreover, the toolpath generated can be

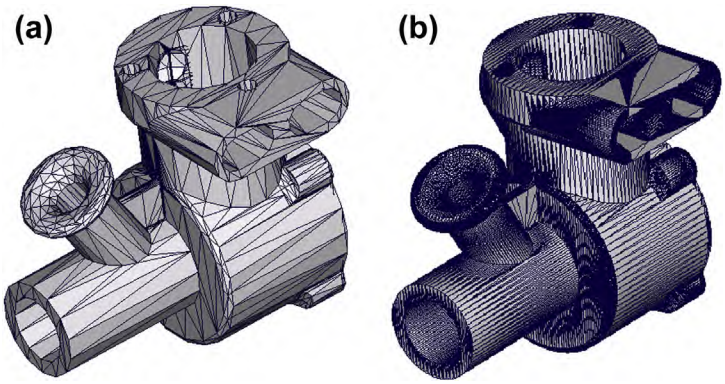


FIGURE 1.15

STL engine case models: (a) coarse and (b) refined.

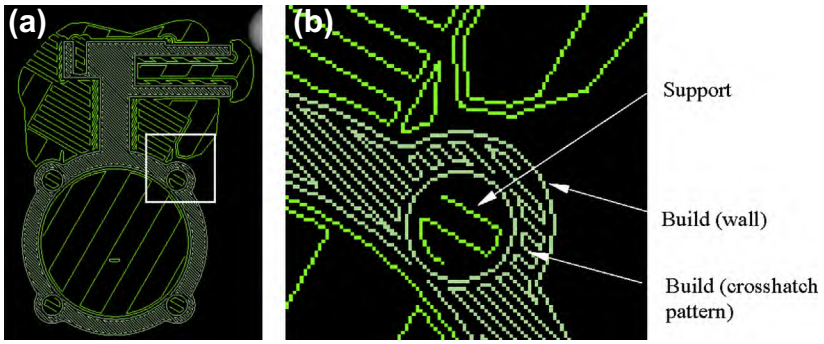


FIGURE 1.16

Crosshatch pattern of a typical cut-out layer: (a) overall and (b) zoom-in area.

converted into CNC codes (M-codes and G-codes) (Chang et al., 1998b; McMahon and Browne, 1998) to fabricate functional parts as well as a die or mold for production.

For example, the cover die of a mechanical part is machined from an 8 in. × 5.25 in. × 2 in. steel block, as shown in Figure 1.17(a). The cutter location data files generated from virtual machining are post-processed into machine control data (MCD)—that is, G- and M-codes, for CNC machining, using post-processor UNCX01.P11 in Pro/MFG. In addition to volume milling and contour surface milling, drilling operations are conducted to create the waterlines. A 3-axis CNC mill, HAAS VF-series (HAAS Automation, Inc., 1996, www.haascnc.com), is employed for fabricating the die for casting the mechanical part (Figure 1.17b).

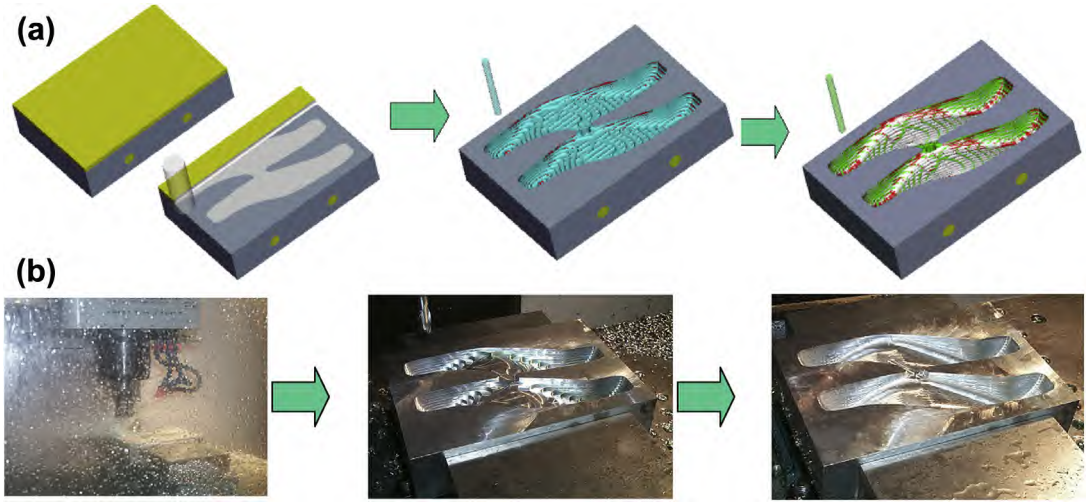


FIGURE 1.17

Cover die machining: (a) virtual and (b) CNC.

1.5 EXAMPLE: SIMPLE AIRPLANE ENGINE

A single-piston, two-stroke, spark-ignition airplane engine (shown in Figure 1.5) is employed to illustrate the e-Design paradigm and tool environment. The cross-functional team is asked to develop a new model of the engine with a 30% increment in both maximum torque and horsepower at 1,215 rpm. The design of the new engine will be carried out at two interrelated levels: system and component. At the system level, the performance measure is the power output; at the component level, the structural integrity and manufacturing cost of each component are analyzed for improvement. Note that only a very brief discussion is provided in this introductory chapter. The computation and modeling details are discussed in Projects S5.3 and P5.3.

1.5.1 SYSTEM-LEVEL DESIGN

Power is proportional to the rotational speed of the crankshaft (N), the swept volume (V_s), and the brake mean effective pressure (P_b) (Taylor, 1985):

$$W_b = P_b V_s N. \tag{1.9}$$

The effective pressure P_b applied on top of the piston depends on, among other factors, the swept volume and the rotational speed of the crankshaft. The pressure is limited by the integrity of the engine structure.

Design variables at the system level include bore diameter (d46:0) and stroke, defined as the distance between the top face of the piston at the bottom and top dead-center positions. In the CAD model, the stroke is twice the crank offset length (d6:6). Both the strike and the connecting rod length (d0:10), as shown in Figure 1.18, affect the performance of the engine. To achieve the requirement for

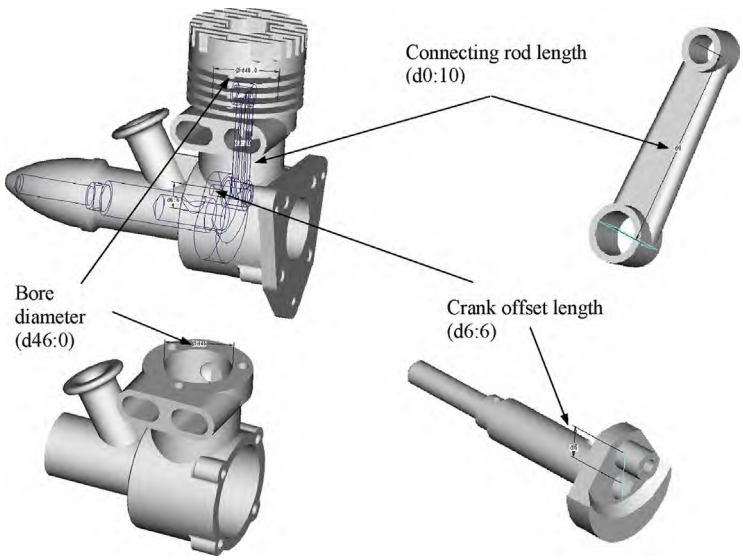


FIGURE 1.18

Engine assembly with design variables at the system level.

Table 1.1 Changes in Design Variables at the System Level

Design Variable	Current Value (in.)	New Value (in.)	Change (in.)	% Change
Bore diameter (d46:0)	1.416	1.6	0.164	11.6
Crank length (d6:6)	0.5833	0.72	0.1567	26.9
Connecting rod length (d0:10)	2.25	2.49	0.24	10.7

system performance, these three design variables are modified as listed in Table 1.1. The design variable values were calculated following theory and practice for internal combustion engines (Taylor, 1985). Details of the computation can be found in Silva (2000).

The solid models of the entire engine are automatically updated and properly assembled using the parametric relations established earlier (refer to Figure 1.6(b)). The change causes P_b to increase from 140 to 180 lbs, so the peak load increases from 400 to 600 lbs. The load magnitude and path applied to the major load-carrying components, such as the connecting rod and crankshaft, are therefore altered. Results from motion analysis show that the system performs well kinematically. Reaction forces applied to the major load-carrying components are computed—for example, for the connecting rod shown in Figure 1.19. The change also affects manufacturing time for some components.

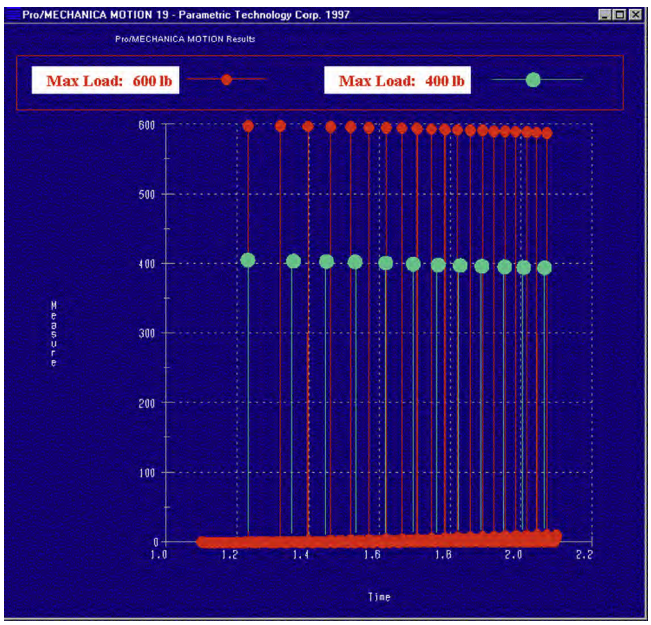


FIGURE 1.19

Dynamic load applied to the connecting rod.

1.5.2 COMPONENT-LEVEL DESIGN

Structural performance is evaluated and redesigned to meet the requirements. In addition, virtual manufacturing is conducted for components with significant design changes. Build materials (volume) and manufacturing times constitute a significant portion of the product cost. In this section, the design of the connecting rod is presented to demonstrate the design decision-making method discussed.

Because of the increased load transmitted through the piston and the increased connecting rod length, the connecting rod can experience buckling failure during combustion. In addition, due to the length change, the natural frequency of the connecting rod may be different. Moreover, load is repeatedly applied to the connecting rod, potentially leading to fatigue failure. Structural FEAs are conducted to evaluate performance of the connecting rod. In addition, virtual manufacturing is carried out to determine the machining cost of the rod.

Because of the increment of the connecting rod length (d0:10) and the magnitude of the external load applied (see Figure 1.20), the rod's maximum von Mises stress increases from 13,600 to 18,850 psi and the buckling load factor decreases from 33 to 7. The first natural frequency is 1,515 Hz. The machining time estimated for the connecting rod is 13.2 minutes using hole-drilling and face-milling operations (shown earlier in Figure 1.9(d)).

1.5.3 DESIGN TRADE-OFF

The design trade-off method discussed in Section 1.3.5 is applied to the components, with significant changes resulting from the system-level design. Only the design trade-off conducted for the connecting rod is presented in this subsection.

Performance measures for the connecting rod, including buckling load factor, fatigue life, natural frequency, volume, and machining costs (time), are brought together for design trade-off. Three design variables, ϕ32, ϕ31, and d7, are identified, as shown in Figure 1.20(b). The objective is to minimize volume and manufacturing time subject to maximum allowable von Mises stress, operating frequency, and minimum allowable buckling load factor. The engine is designed to work at 21 kHz, and the minimum allowable buckling load factor for the connecting rod is assumed to be 10.

Sensitivity coefficients for performance and cost measures with respect to design variables are calculated (refer to Figure 1.11) using the finite difference method. Design trade-offs are conducted

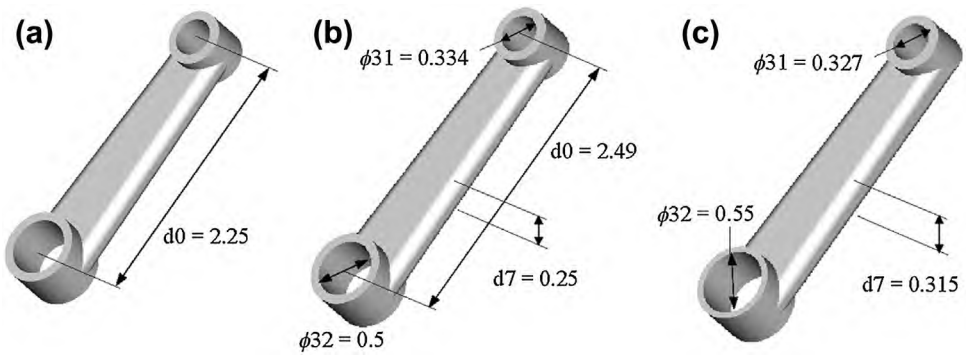


FIGURE 1.20

Engine connecting rod: (a) original design; (b) changes at the system level; (c) changes at the component level.

Table 1.2 Changes in Design Variables at the Component Level

Design Variable	Current Value (in.)	New Value (in.)	% Change
Diameter of the large hole (ϕ32)	0.50	0.55	10
Diameter of the small hole (ϕ31)	0.334	0.32728	−2.01
Thickness (d7)	0.25	0.31484	25.9

Table 1.3 Changes in Performance Measures at the Component Level

Performance Measure	Current Value	New Value	% Change
VM stress	18.9 ksi	10.5 ksi	−44.4
Buckling load factor	7.1	14.2	100
Volume	0.438813 $in.^3$	0.5488 $in.^3$	25.1
Machining time	13.2 min	13.2 min	0
Natural frequency	1515 Hz	1840 Hz	21.5

followed by a what-if study. When a satisfactory design is found, the solid model of the rod is updated for performance evaluation and virtual manufacturing. This process is repeated twice when all the requirements are met. The design change is summarized in Tables 1.2 and 1.3, which show that the machining time is maintained and a small volume increment is needed to achieve the required performance.

1.5.4 RAPID PROTOTYPING

When the design is finalized through virtual prototyping, rapid prototyping is used to fabricate a physical prototype of the engine, as shown in Figure 1.21. The prototype can be used for design verification as well as tolerance and assembly checking.

1.6 EXAMPLE: HIGH-MOBILITY MULTIPURPOSE WHEELED VEHICLE

The overall objective of the high-mobility multipurpose wheeled vehicle (HMMWV) design is to ensure that the vehicle's suspension is durable and reliable after accommodating an additional armor loading of 2,900 lb. A design scenario using a hierarchical product model (see Figure 1.10) that evolves during the design process is presented in this section. Only a brief discussion is presented. More details can be found in Chang et al. (1998a).

In the preliminary design stage, vehicle motion is simulated and design changes are performed to improve the vehicle's gross motion. At this stage, the dynamic behavior of the HMMWV's suspension is simulated and designed. The specific objectives of the preliminary design are to avoid the problem of metal-to-metal contact in the shock absorber due to added armor load, and to improve the driver's comfort by reducing vertical acceleration at the HMMWV driver's seat.

FIGURE 1.21

Physical prototypes of engine parts.

By modifying the spring constant to improve the HMMWV suspension design at the preliminary design stage, the load path generated in HMMWV dynamics simulation is affected in the suspension unit. In the detailed design stage, the objective is to assess and redesign the durability, reliability, and structural performance of selected suspension components affected by the added armor load that results in changes in load path and magnitude.

Note that only a very brief discussion is provided in this introductory chapter. The computation and modeling details are discussed in later chapters.

1.6.1 HIERARCHICAL PRODUCT MODEL

In this case, a hierarchical product model is employed to support the HMMWV's design. In all models, nonsuspension parts, such as instrument panel, seats, and lights, are not modeled. Important vehicle components, such as engine and transmission, are modeled using engineering parameters without depending on CAD representation. A low-fidelity CAD model consisting of 18 parts (Figure 1.22) is created using Pro/ENGINEER to support the preliminary design. This model has accurate joint definition and fairly accurate mass property, but less accurate geometry. The goal of the low-fidelity model is to support vehicle dynamic simulation. It is created using substantially less effort compared to that required for the detailed model.

FIGURE 1.22

HMMWV CAD model for preliminary design.

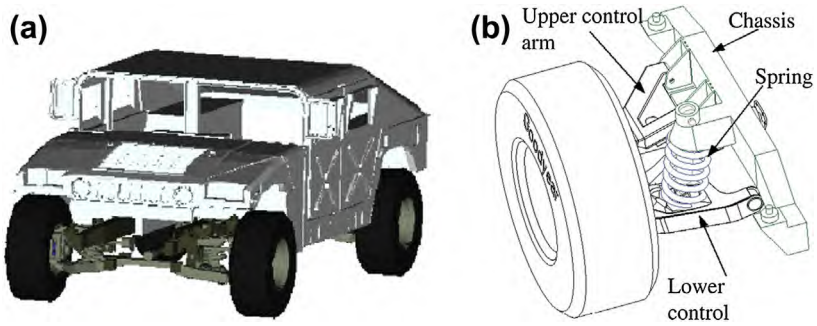


FIGURE 1.23

HMMWV CAD model for detail design.

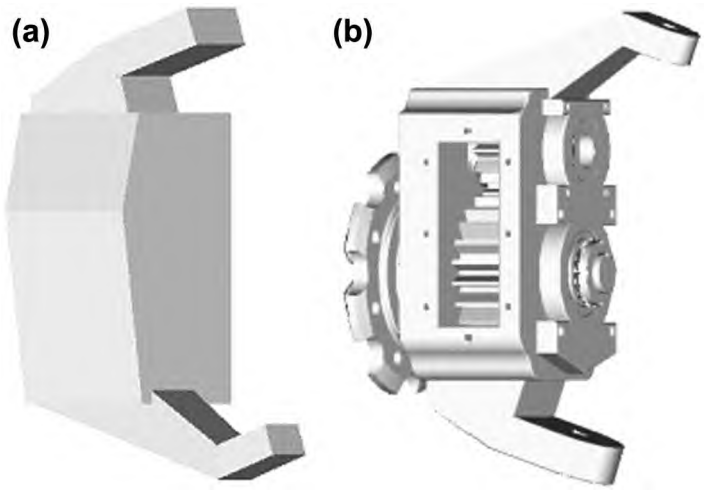


FIGURE 1.24

HMMWV gear hub assembly models: (a) preliminary and (b) detailed.

The detailed product model, consisting of more than 200 parts and assemblies (Figure 1.23), is created to support the detail design of suspension components. The detailed model is derived from the preliminary model by (1) breaking an entity into more parts and assemblies (e.g., the gear hub assembly, shown in Figure 1.24) to simulate and design detailed parts, and (2) refining the geometry of mechanical components to support structural FEA (e.g., the lower control arm, shown in Figure 1.25).

1.6.2 PRELIMINARY DESIGN

The HMMWV is driven repeatedly on a virtual proving ground, as shown in Figure 1.26, with a constant speed of 20 MPH for a period of 23 seconds. A dynamic simulation model, shown in

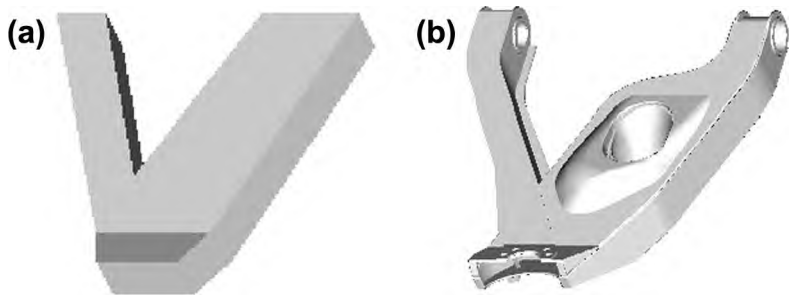


FIGURE 1.25

HMMWV lower control arm models: (a) preliminary and (b) detailed.

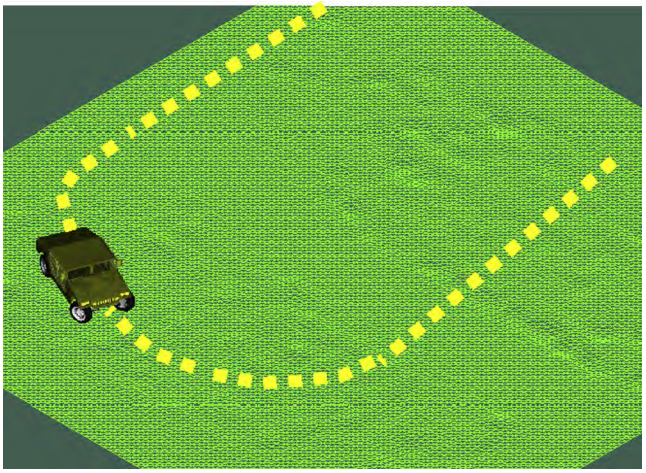


FIGURE 1.26

HMMWV dynamic simulation.

Figure 1.27, is first derived from the low-fidelity CAD solid model of the HMMWV (refer to Figure 1.22). A more in-depth discussion of the HMMWV vehicle dynamic model is provided in Chapter 8, Motion Analysis.

Using DADS, severe metal-to-metal contact is identified within the shock absorber, caused by the added armor load and rough driving conditions, as shown in Figure 1.28. The spring constant is adjusted to avoid any contact problems; it is increased in proportion to the mass increment of the added armor to maintain the vehicle's natural frequency. This design change not only eliminates the contact problem (see Figure 1.28) but also reduces the amplitude of vertical acceleration at the driver's seat, which improves driving comfort (see Figure 1.29). However, the change alters the load path in the components of the suspension system—for example, the shock absorber force acting on the control arm, as shown in Figure 1.30, increases about 75%.

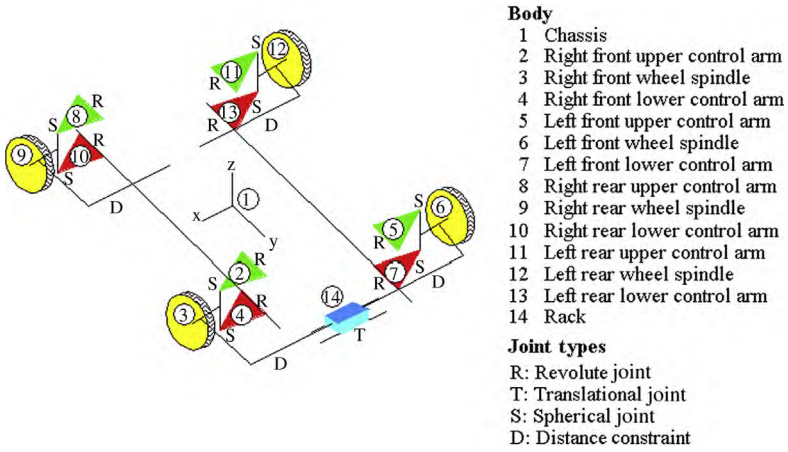


FIGURE 1.27

HMMWV dynamic model.

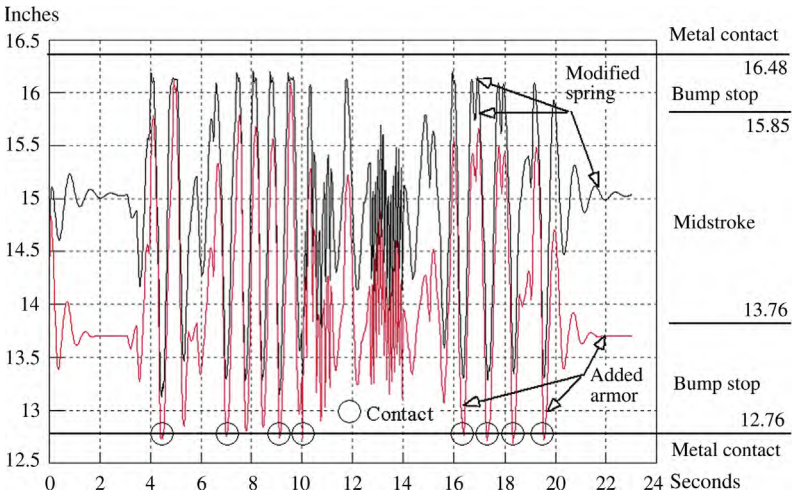


FIGURE 1.28

Shock absorber operation distance (in inches).

1.6.3 DETAILED DESIGN

Simulations are carried out for fatigue, vibration, and buckling of the lower control arm (Figure 1.30); reliability of gears in the gear hub assembly (refer to Figure 1.24(b)); the spring of the shock absorber (see Figure 1.23); and the bearings of the control arm (see Figure 1.30).

Using ANSYS, the first natural frequency of the lower control arm is obtained as 64 Hz, which is far away from vehicle vibration frequency, eliminating concern about resonance. The buckling load

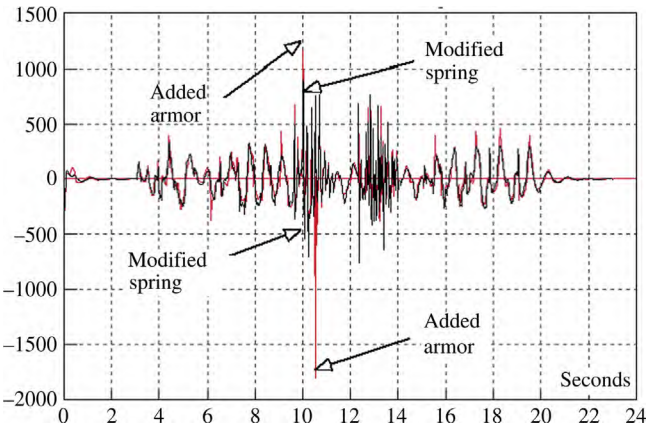


FIGURE 1.29

HMMWV driver seat vertical accelerations (in./sec^2).

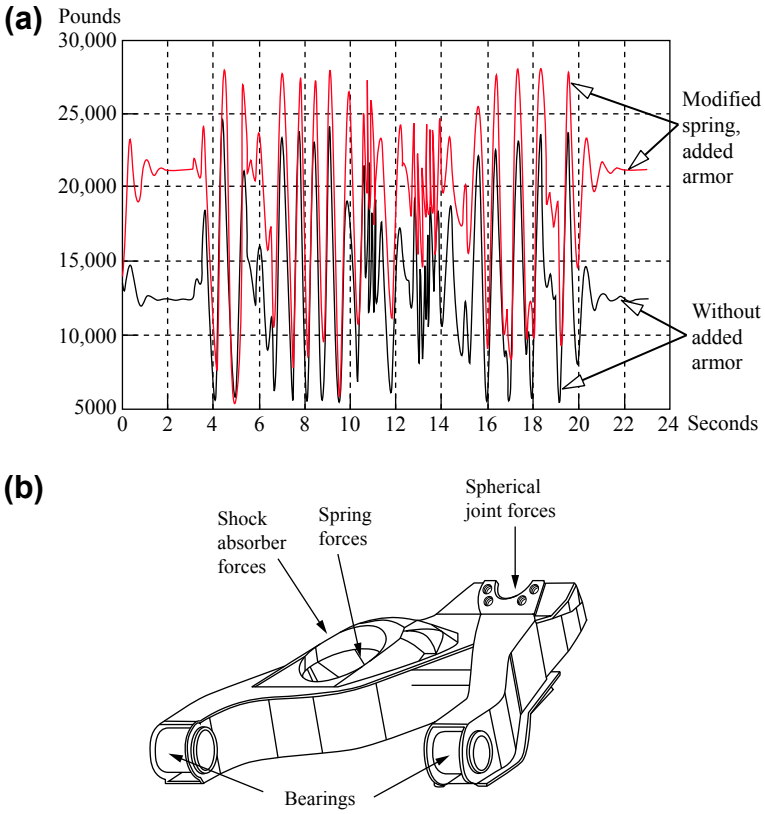


FIGURE 1.30

History of shock absorber forces (lbs): (a) force history with and without added armor load, (b) locations of force application.

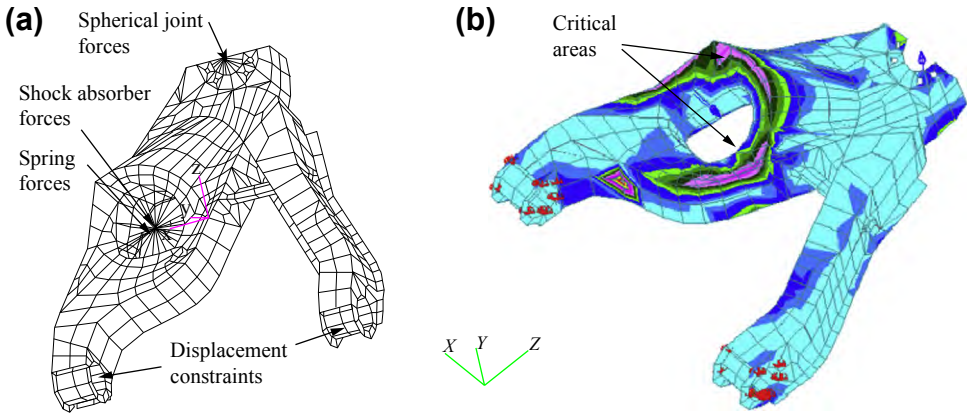


FIGURE 1.31

HMMWV lower control arm models: (a) finite element and (b) fatigue life prediction.

factor is analyzed using the peak load at time 10.05 seconds in the 23-second simulation period. The result shows that the control arm will not buckle even under the most severe load. Therefore, the current design is acceptable as far as buckling and resonance of the lower control arm are concerned.

Results obtained from fatigue analyses show that fatigue life (crack initiation) of the lower control arm degrades significantly—for example, from 6.61E+09 to 1.79E+07 blocks (one block is 20 seconds) at critical areas (see Figure 1.31(b))—because of the additional armor load. Therefore, the design must be altered to improve control arm durability. Reliability of the bearing, gear, and spring at a 99% fatigue failure rate is 2.18E+07, 3.36E+06, and 1.27E+02 blocks, respectively. Note that the fatigue life of the spring at the required reliability is not desirable.

1.6.4 DESIGN TRADE-OFF

Eleven design variables, including geometric dimensions (d1 and d2 in Figure 1.32(a)), material property (cyclic strength coefficient K' of the lower control arm), and thickness of the control arm sheet metal (t1 to t7 in Figure 1.32(b)) are defined to support design modification.

A global design trade-off that involves changes in more than one component is conducted first. Geometric design variables d1 and d2 are modified to reduce loads applied to the control arm, bearing, spring, and gears in the gear hub so that the durability and reliability of these components can be improved. Changes in d1 and d2 affect not only the lower control arm but also the upper control arm and the chassis frame. Sensitivity coefficients of loads at discretized time steps (a total of 10 selected time steps) with respect to parameters d1 and d2 are calculated using a finite difference method. Sensitivity coefficients can be displayed in bar charts (see Figure 1.33(a)) to guide design modifications. A what-if study is carried out with a design perturbation of 0.6 and 0.3 in. for d1 and d2, respectively, to obtain a reduction in loads. An example of the what-if results is shown in Figure 1.33(b).

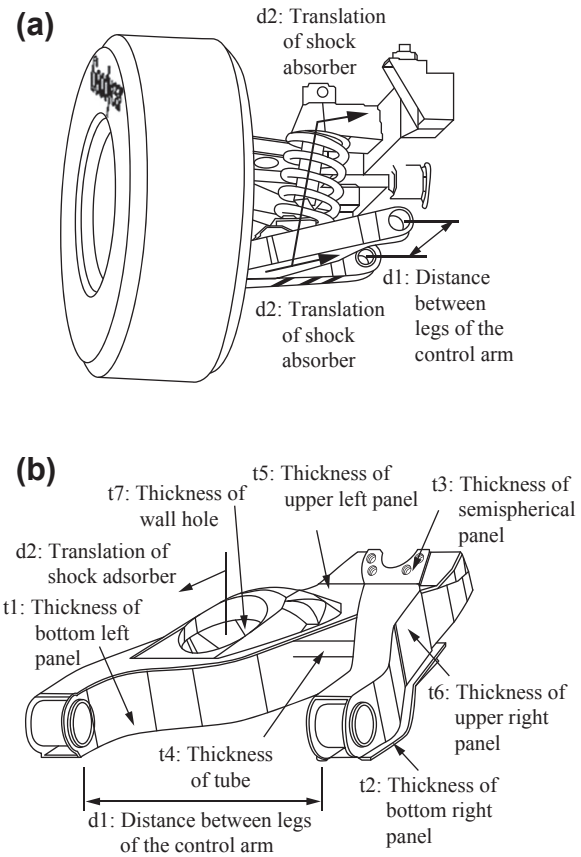


FIGURE 1.32

Design variables defined for the control arm: (a) suspension geometric dimensions and (b) thickness dimensions.

A local design trade-off that involves design variables of a single component is carried out for the lower control arm. Thickness design variables t1 to t7 and the material design parameter K' are modified to increase the control arm's fatigue life. Fatigue life at ten nodes of its finite element model in the critical area is measured. Sensitivity coefficients of control arm fatigue life at these nodes with respect to the thickness and material parameters are calculated. A design trade-off method using a QP algorithm is employed because of the large number of design variables and performance measures involved. An improved design obtained shows that with a 0.6% weight increment, fatigue life at the critical area increases about ten times: from 1.79 E+07 to 1.68 E+08 blocks.

A dynamic simulation is performed again with the detailed model and modified design to ensure that the metal contact problem, encountered in the preliminary design stage, is eliminated as a result of

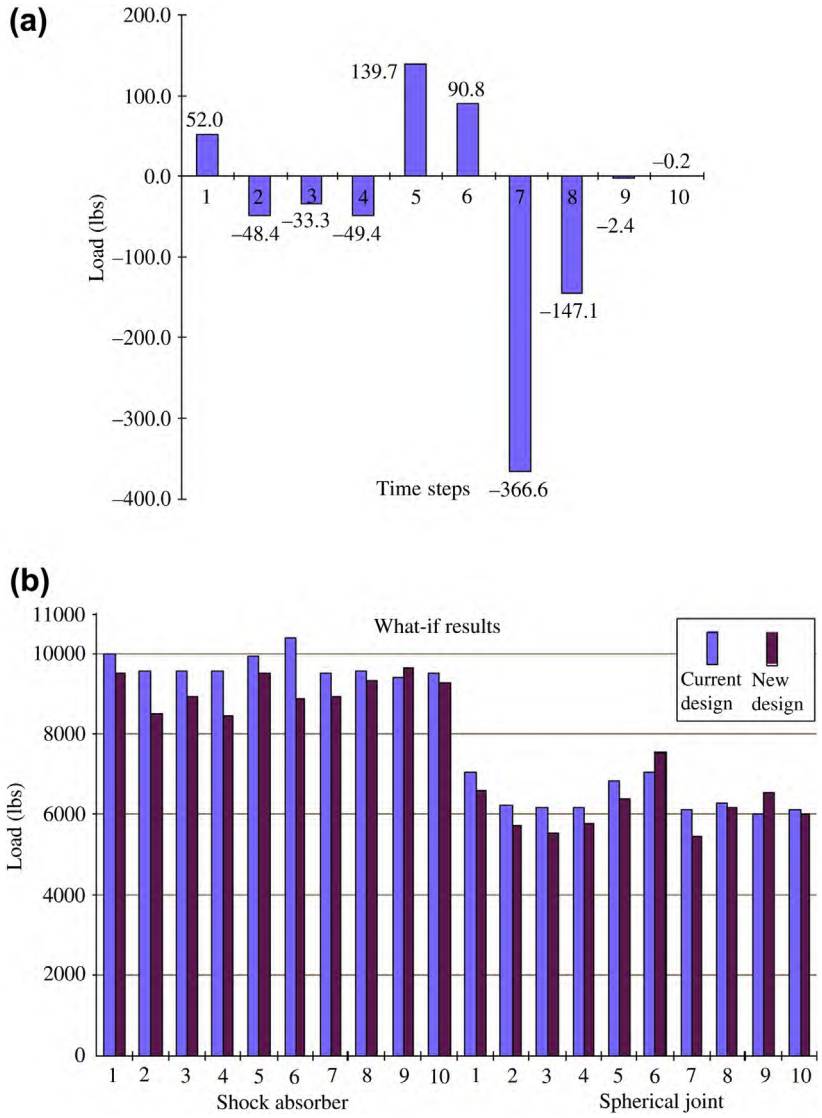


FIGURE 1.33

Sensitivity and what-if study of loads (a) sensitivity of loads on the spherical joint of control arm w.r.t. d2 at 10 time steps and (b) what-if study of loads on the shock and spherical joint due to design changes in the design variables d1 and d2.

model refinement and design changes in the detailed design stage. The global design trade-off reduces the load applied to the shock absorber spring. This reduction significantly increases the spring fatigue life to the desired level.

1.7 SUMMARY

In this chapter, the e-Design paradigm and software tool environment were discussed. The e-Design paradigm employs virtual prototyping for product design and rapid prototyping and computer numerical control (CNC) for fabricating physical prototypes of a design for design verification and functional tests. The e-Design paradigm offers three unique features:

- The VP technique, which simulates product performance, reliability, and manufacturing costs; and brings these measures to design.
- A systematic and quantitative method for design decision making for the parameterized product in solid model forms.
- RP and CNC for fabricating prototypes of the design that help verify product design and bring marketing personnel and potential customers into the design loop.

The e-Design approach holds potential for shortening the overall product development cycle, improving product quality, and reducing product costs. With intensive knowledge of the product gained from simulations, better design decisions can be made, thereby overcoming what is known as the *design paradox*. With the advancement of computer simulations, more hardware tests can be replaced by them, reducing cost and shortening product development time. Manufacturing-related issues can be largely addressed through virtual manufacturing in early design stages. Moreover, manufacturing process planning conducted in virtual manufacturing streamlines the production process.

QUESTIONS AND EXERCISES

1.1. In this assignment, you are asked to search and review articles (such as in *Mechanical Engineering* magazine) that document successful stories in industry that involve employing the e-Design paradigm and/or employing CAD/CAE/CAM technology for product design.
- Briefly summarize the company's history and its main products.
- Briefly summarize the approach and process that the company adopted for product development in the past.
- Why must the company make changes? List a few factors.
- Which approach and process does the company currently employ?
- What is the impact of the changes to the company?
- In which journal, magazine, or website was the article published?

1.2. In this chapter we briefly discussed rapid prototyping technology and the Dimension 1200 sst machine. The sst uses fused deposition manufacturing technology for support of layer manufacturing. Search and review articles to understand the FDM technology and machines that employ such technology other than the Dimension series.

REFERENCES

Anderson, D.M., 1990. Design for Manufacturability: Optimizing Cost, Quality, and Time to Market. CIM Press.

Arora, J.S., 1989. Introduction to Optimal Design. McGraw-Hill.

Boothroyd, G., Dewhurst, P., Knight, W., 1994. Product Design for Manufacturing and Assembly. Marcel Dekker.

Chang, K.H., Choi, K.K., Wang, J., Tsai, C.S., Hardee, E., 1998a. A multi-level product model for simulation-based design of mechanical systems. Concurrent Engineering Research and Application (CERA) Journal 6 (2), 131–144.

Chang, K.H., Yu, X., Choi, K.K., 1997. Shape design sensitivity analysis and optimization for structural durability. International Journal of Numerical Methods in Engineering 40, 1719–1743.

Chang, T.C., Wysk, R.A., Wang, H.P., 1998b. Computer-Aided Manufacturing, second ed. Prentice Hall.

Chen, J.S., Pan, C., Wu, T.C., 1997. Large deformation analysis of rubber based on a reproducing kernel particle method. Computational Mechanics 19, 153–168.

Chua, C.K., Leong, K.F., 1998. Rapid Prototyping: Principles and Applications in Manufacturing. John Wiley.

Ertas, A., Jones, J.C., 1993. The Engineering Design Process. John Wiley.

Fang, X.D., Jawahir, I.S., 1996. A hybrid algorithm for predicting chip form/chip breakability in machining. International Journal of Machine Tools and Manufacture 36 (10), 1093–1107.

Gonzalez, A.J., Dankel, D.D., 1993. The Engineering of Knowledge-Based Systems, Theory and Practice. Prentice Hall.

Grindeanu, I., Choi, K.K., Chen, J.S., Chang, K.H., 1999. Design sensitivity analysis and optimization of hyperelastic structures using a meshless method. AIAA Journal 37 (8), 990–997.

HAAS Automation Inc, 1996. VF Series Operations Manual.

Hallquist, J.O., 2006. LS-DYNA3D Theory Manual. Livermore Software Technology Corp.

Haug, E.J., Choi, K.K., Komkov, V., 1986. Design Sensitivity Analysis of Structural Systems. Academic Press.

Haug, E.J., 1989. Computer-Aided Kinematics and Dynamics of Mechanical Systems, vol. I: Basic Methods. Allyn and Bacon.

Jacobs, P.F., 1994. StereoLithography and Other RP&M Technologies. ASME Press.

Kalpakjian, S., 1992. Manufacturing Engineering and Technology, second ed. Addison-Wesley.

Kane, T.R., Levinson, D.A., 1985. Dynamics: Theory and Applications. McGraw-Hill.

Lee, W., 1999. Principles of CAD/CAM/CAE Systems. Addison-Wesley Longman.

Madsen, H.O., Krenk, S., Lind, N.C., 1986. Methods of Structural Safety. Prentice Hall.

McMahon, C., Browne, J., 1998. CADCAM, second ed. Addison-Wesley.

Moës, N., Gravouil, A., Belytschko, T., 2002. Nonplanar 3D crack growth by the extended finite element and level sets. Part I: Mechanical model. International Journal for Numerical Methods in Engineering 53 (11), 2549–2568.

Numerical Integration Technologies, 1998. SYSNOISE 5.0.

Prasad, B., 1996. Concurrent Engineering Fundamentals, vols. I and II: Integrated Product and Process Organization. Prentice Hall.

Silva, J., 2000. Concurrent Design and Manufacturing for Mechanical Systems. MS thesis University of Oklahoma.

Szabó, B., Babuška, I., 1991. Finite Element Analysis. John Wiley.

Taylor, C., 1985. The Thermal-Combustion Engine in Theory and Practice, vol. I: Thermodynamics, Fluid Flow, Performance, second ed. MIT Press.

Tsai, C.S., Chang, K.H., Wang, J., 1995. Integration infrastructure for a simulation-based design environment. In: Proceedings, Computers in Engineering Conference and the Engineering Data Symposium. ASME Design Theory and Methodology Conference.

Ullman, D.G., 1992. The Mechanical Design Process. McGraw-Hill.
Wu, Y.T., Wirsching, P.H., 1984. Advanced reliability method for fatigue analysis. Journal of Engineering Mechanics 110, 536–563.
Yu, X., Chang, K.H., Choi, K.K., 1998. Probabilistic structural durability prediction. AIAA Journal 36 (4), 628–637.
Yu, X., Choi, K.K., Chang, K.H., 1997. A mixed design approach for probabilistic structural durability. Journal of Structural Optimization 14, 81–90.
Zeid, I., 1991. CAD/CAM Theory and Practice. McGraw-Hill.

CHAPTER

GEOMETRIC MODELING

2

CHAPTER OUTLINE

2.1 Introduction 43
2.2 Parametric Curves 44
2.2.1 Straight Line 46
2.2.2 Quadratic Curves 47
2.2.2.1 Spline Curve—Three Points 48
2.2.2.2 Two Points and a Vector 50
2.2.2.3 Bézier Curve 52
2.2.3 Cubic Curves 56
2.2.3.1 Spline Curve—Four Points 56
2.2.3.2 Hermit Cubic Curve (Two End Points and Two End Vectors) 59
2.2.3.3 Bézier Curve 62
2.2.4 Continuities 63
2.2.5 B-Spline Curves 64
2.2.5.1 Nonuniform B-Spline Curves 64
2.2.5.2 Uniform B-Spline Curves 69
2.2.5.3 Closed Uniform B-Spline Curves 71
2.2.6 NURB Curves 75
2.3 Parametric Surfaces 76
2.3.1 Parametric Representation 77
2.3.1.1 Bicubic Surface Patch 77
2.3.1.2 16-Point Format 78
2.3.1.3 Coons Patch 79
2.3.1.4 Bézier Surface 81
2.3.2 B-Spline Surface 82
2.4 CAD-Generated Surfaces 84
2.4.1 Cylindrical Surfaces 84
2.4.2 Ruled Surfaces 87
2.4.3 Loft (or Blend) Surfaces 89
2.4.4 Revolved Surfaces 92
2.4.5 Sweep Surfaces 96
2.5 Geometric Transformations 100
2.5.1 Homogeneous Coordinates 100

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00002-8
Copyright © 2015 Elsevier Inc. All rights reserved.

2.5.2 Scaling 102
2.5.3 Translation 103
2.5.4 Rotations 104
2.5.5 Composite Transformations 105
2.6 Case Studies 108
2.6.1 Curve Fitting and Surface Skinning 108
2.6.1.1 Curve Fitting 108
2.6.1.2 Surface Skinning 110
2.6.2 Engineering Applications 111
2.7 Summary 114
Appendix 2A: Basis Functions of B-spline Curves and Surfaces 114
Appendix 2B: Representing Conics with Quadratic NURB Curves 116
Questions and Exercises 120
References 124

Virtual prototyping is becoming a cornerstone in modern product development. In e-Design, product design is first realized in computer-aided design (CAD) solid model form as parts and assemblies. The CAD solid model describes a geometric shape and physical properties that are essential for support of the product design, particularly product performance evaluation, virtual manufacturing, and cost estimating. The CAD solid model must be also properly parameterized in order for the design team to explore design alternatives for better product performance and hopefully less cost.

Most CAD software employs a geometric modeling kernel, such as Parasolid or ACIS, which is the library of core mathematical functions that define and store three-dimensional (3D) solid objects, for support of product modeling. In solid modeling, geometry is formed as a combination of constituent solid objects (more specifically solid features), which are created mostly by sketching a two-dimensional (2D) profile, composed of line or curve entities, and protruding the profile for a solid object. While protruding the profile for a solid object, the trace of the line or curve entities forms boundary surfaces that wrap the solid object.

Therefore, before getting into the solid modeling and CAD theories, it is indispensable for readers to acquire a fundamental knowledge in curves and surfaces, which is often referred to as geometric modeling. This chapter focuses on introducing basic topics in geometric modeling, including curves, surfaces, and geometric transformations that are required for transforming geometric entities to meet specific needs. We assume readers who have used CAD software for creating solid models, but have no or little background in geometric modeling (and CAD theory). Therefore, instead of focusing on the use of CAD software, we focus more on understanding the selected topics in geometric modeling (and CAD theories in the next chapter). These topics are essential and relevant for readers to gain more in-depth understanding in behind-the-scenes operations while using CAD software. For a more comprehensive discussion on geometric modeling, readers are referred to excellent books, such as Mortenson (2006).

In this chapter, we provide fairly thorough discussions on parametric representations for the basic curves and surfaces that are widely employed in geometric modeling. Such curves and surfaces include Hermit cubic curve, Coons patch, Bézier curves and surfaces, B-spline curves and surfaces, and nonuniform rational B-spline (NURB) curves and surfaces that are considered to be the most versatile and general form for representing geometric entities. We also discuss surfaces generated by protruding sketch profiles in CAD, such as cylindrical, ruled, revolved, sweep, and loft. In addition, we discuss

geometric transformations, including scaling, translation, and rotation, which are commonly employed to manipulate geometric entities to meet specific modeling needs.

This chapter is essentially a prelude to the subjects that are more directly relevant to product design modeling, such as CAD theories, parameterization, product data exchanges, and so forth, which are discussed next. Overall, the objective of this chapter is to provide an introduction to the parametric representations for curves and surfaces that help readers understand how the geometric models are defined mathematically.

2.1 INTRODUCTION

Geometric modeling is in general considered as a branch of applied mathematics and computational geometry that studies methods and algorithms for the mathematical description of shapes that represent geometry of objects. Geometric modeling has been an important and interesting subject for many years from the purely mathematical and computer science viewpoint, and also from the standpoint of engineering and various other applications, such as CAD/CAM (computer-aided design/manufacturing), entertainment, animation, and multimedia. Our interest is certainly in CAD/CAM, especially product modeling in CAD, in which objects are constructed by first creating curves and surfaces before reaching a solid model. Geometric modeling is indeed the backbone of a CAD system. It deserves our attention because understanding geometric modeling technique is a key step in learning CAD, especially for understanding its behind-the-scenes operations. With a solid background in geometric modeling, it should be easier for you to learn to use CAD software, avoid potential pitfalls, and be able to diagnose problems encountered in solid modeling.

Before getting into the discussion, a few points must be kept in mind. First, geometric modeling is only a means, not the goal, in engineering. In engineering design, analysis, and manufacturing, product geometry with an adequate level of detail must be available. This is especially true in e-Design, in which product design is refined with significant geometric detail as the development process gets into the later stages. Geometric modeling provides the fundamental means in representing design with the needed level of detail that supports engineering design in various stages.

This chapter attempts to help you understand how modeling is carried out on a computer. To support creating geometric models on a computer, computational algorithms must be implemented on computer systems. Traditional mathematical methods learned in high school mathematical courses are based upon continuous functions, and computer systems do not generally work this way; they are discrete beasts. Thus, in the early 1970s, it was recognized that we could not represent curves by a general continuous function but must represent them as discrete entities. It is the reduction of these continuously defined mathematical objects to a more discrete representation that has motivated the field of geometric modeling. In any case, the mathematical representations of curves and surfaces are essential, to say the least.

This chapter focuses on the discussion of mathematical representations of curves and surfaces, as well as the transformation of these geometric entities for various modeling purposes, such as scaling, rotation, and translation. We provide fairly thorough discussions on parametric representations for the most basic and popular curves and surfaces in Sections 2.2 and 2.3, respectively. Such curves and surfaces include the Hermit cubic curve, Coons patch, Bézier curves and surfaces, B-spline curves and surfaces, and NURB curves and surfaces, which are considered to be the most versatile and general forms for representing geometric entities. In addition to the basic surfaces, we include surfaces

generated by CAD in Section 2.4. In CAD, we sketch a profile and protrude it for a surface (or solid). The protrusion capabilities commonly available in CAD include extrusion, blend (or loft), revolve, and sweep. The mathematical representations of these surfaces are presented together with Matlab scripts that graph them for visualization. We also include the discussion of geometric transformations in Section 2.5. Finally, in Section 2.6, we include case studies that showcase the applications of geometric modeling to practical applications, which include curve fitting and surface skinning techniques, and applications of the techniques to practical modeling examples.

2.2 PARAMETRIC CURVES

Typically, in high school mathematics or trigonometry, a curve is presented as a graph of a function $f(x)$, as shown in Figure 2.1(a). As x is varied, $y = f(x)$ is computed by the function f, and the pair of coordinates (x,y) sweeps out the curve. This is called the *explicit* form of the curve representation.

In addition to the explicit form, a geometric curve can be presented in an *implicit* form as $F(x,y) = 0$. For example, a circle of radius r and center at (a,b) in a Cartesian coordinate system x–y, shown in Figure 2.1(b), can be written as

$$F(x, y) = (x - a)^2 + (y - b)^2 - r^2 = 0. \tag{2.1}$$

In addition to a circle, a number of basic Conic curves shown in Figure 2.2 are commonly found in describing geometry of mechanical parts. They are all quadratic functions in two variables represented in the Cartesian coordinate system as an implicit form. As shown in Figure 2.2, the graph of them is always a conic section.

Is CAD using such mathematical forms, either the explicit form $y = f(x)$ or the implicit form $F(x,y) = 0$, to represent curves (or surfaces) and carry out computations internally? The answer is generally no.

From a design perspective, such forms are inadequate in several ways. If we take the circle shown in Figure 2.1(b) as an example, the implicit form is inconvenient for computing points on the curve. For example, consider a circle defined as $F(x,y) = (x - 1)^2 + (y - 2)^2 - 1^2 = 0$. If one chooses $x = 3$, then $(y - 2)^2 + 3 = 0$ does not have a real solution for y, implying that the vertical line $x = 3$ does not intersect with the circle. Also, the implicit form may not be a single-valued function when there is a solution. As shown in Figure 2.3(a), the curve is not single-valued along lines that are inside the circle and are parallel to the y axis. Also, it is cumbersome to transform, such as to rotate geometric curves

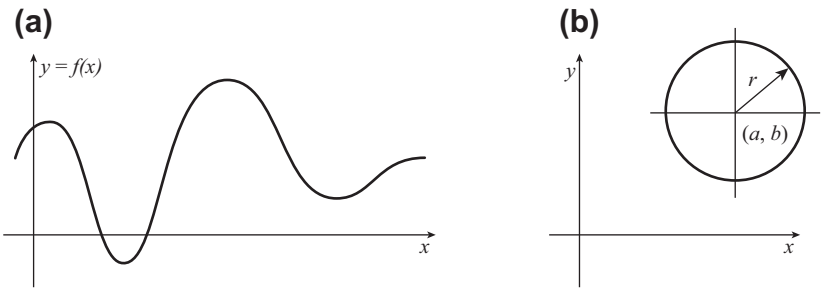
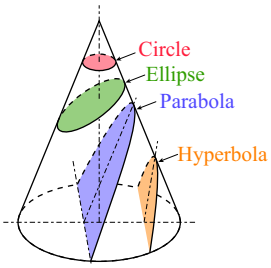


FIGURE 2.1

A curve representation. (a) An explicit function $y = f(x)$. (b) An implicit form $F(x,y) = 0$.



Conic section	Equation
Circle	$x^2 + y^2 = a^2$
Ellipse	$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$
Parabola	$y^2 = 4ax$
Hyperbola	$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$

FIGURE 2.2

Conic curve representations.

represented in such forms. For example, as shown in Figure 2.3(b), writing an explicit or implicit function for the 90°-circular arc that is rotated $a - \alpha$ angle along the y-axis is not straightforward, to say the least.

CAD relies on parametric forms to describe curves and surfaces. What is a parametric curve? Is the equation we are familiar with, $(x-a)^2 + (y-b)^2 = r^2$, representing a circle shown in Figure 2.1(b), a parametric curve? The answer is no. What about representing the same circle in a polar coordinate system? For example, a circle of radius r and center point (a,b) can be written in a polar coordinate form as:

$$\begin{aligned} x &= a + \mathrm{r}\cos\theta \\ y &= b + r\sin\theta, \quad \text{and} \quad \theta \in [0, 2\pi] \end{aligned} \tag{2.2}$$

Is Eq. 2.2 parametric? Yes, it is one of the parametric forms, in which x and y are decoupled and are separately represented in respective trigonometric functions in terms of the common parameter θ. In this case, the angle varying between 0 and 2π.

In general, a parametric curve that lies on an x–y plane is defined by two functions, $x(u)$ and $y(u)$, which use the parameter u. $x(u)$ and $y(u)$ are coordinate functions since their values represent the coordinates of points on the curve. As u varies, the coordinates $(x(u), y(u))$ sweep out the curve.

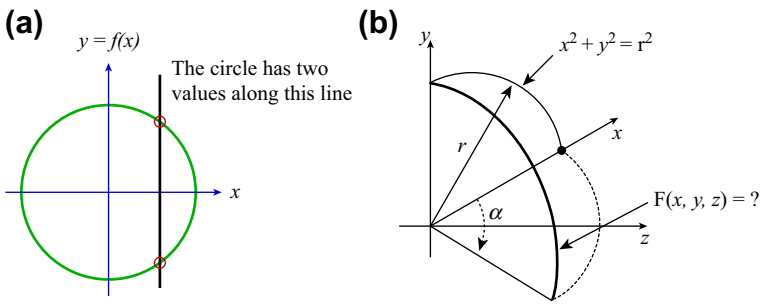


FIGURE 2.3

A circle of radius r and center at $(a,b) = (0,0)$. (a) In a Cartesian coordinate system. (b) In a polar coordinate system.

In general, CAD deals primarily with polynomial or rational functions (made by dividing one polynomial by another) and less on trigonometric functions like those in Eq. 2.2. For example, the circle can also be given by allowing u to vary from $-\infty$ to $+\infty$ in the following functions:

$$\begin{aligned} x(u) &= \frac{2u}{1+u^2} \\ y(u) &= \frac{1-u^2}{1+u^2} \end{aligned} \tag{2.3}$$

Both Eqs 2.2 and 2.3 yield circles, so how do they differ? It is the curve parameterization. The motion of the point $(x(u), y(u))$ is different, even if the paths (the circles) are the same.

A good physical model for parametric curves is that of a moving particle. The parameter u represents time. At any time u, the position of the particle is $(x(u), y(u))$. Two paths (or parametric curves) may be identical even though the motion (or parameterization) is different.

In general, a planar or spatial curve $\mathbf{P}(u)$ can be represented in parametric forms, respectively, as follows:

$$\begin{aligned} \mathbf{P}(u) &= \left[\mathbf{P}_x(u), \quad \mathbf{P}_y(u)\right]_{1\times 2}, \quad u \in [0,1], \text{ and} \\ \mathbf{P}(u) &= \left[\mathbf{P}_x(u), \quad \mathbf{P}_y(u), \quad \mathbf{P}_z(u)\right]_{1\times 3}, \quad u \in [0,1] \end{aligned} \tag{2.4}$$

where u is the parameter, usually in [0,1]. Parametric curves are suitable for modeling curves in CAD. As shown in Eq. 2.4, coordinates of the curves x and y (and z) are decoupled and represented independently by their respective functions, usually explicit, in terms of a single parameter u. When a u value is specified, the coordinates of the curve can always be evaluated using Eq. 2.4. Also, the curve transformations can be easily taken care of by transforming characteristic points of the curve instead of the functions, which will be discussed more in Section 2.5.

In this section, we discuss parametric curves, both polynomial and rational. We will start with a simple straight line, followed by quadratic curves, cubic curves, B-splines, and then rational curves. We assume spatial curves. Planar curves can be easily obtained by removing $\mathbf{P}_z(u)$ (the z-component) of the curve equations. Some of the detailed derivations are either left as exercises or presented in appendices.

2.2.1 STRAIGHT LINE

A straight line shown in Figure 2.4 can be defined in a parametric form as a linear function of u as

$$\mathbf{P}(u) = \left[\mathbf{P}_x(u), \quad \mathbf{P}_y(u), \quad \mathbf{P}_z(u)\right]_{1\times 3} = (1-u)\mathbf{P}_0 + u\mathbf{P}_1, \quad u \in [0,1] \tag{2.5}$$

where $\mathbf{P}_0 = [\mathbf{P}_{0x}, \mathbf{P}_{0y}, \mathbf{P}_{0z}]_{1\times 3}$ and $\mathbf{P}_1 = [\mathbf{P}_{1x}, \mathbf{P}_{1y}, \mathbf{P}_{1z}]_{1\times 3}$ are the start and end points of the line, respectively. When u is 0, $\mathbf{P}(u) = \mathbf{P}(0) = \mathbf{P}_0$, and when $u = 1$, $\mathbf{P}(u) = \mathbf{P}(1) = \mathbf{P}_1$.

Note that Eq. 2.5 can be derived in a more formal way. For a straight line, its coordinates can be represented by a liner function in u, in which $u \in [0,1]$:

$$\begin{aligned} \mathbf{P}_x(u) &= a_{1x}u + a_{0x} \\ \mathbf{P}_y(u) &= a_{1y}u + a_{0y} \\ \mathbf{P}_z(u) &= a_{1z}u + a_{0z} \end{aligned} \tag{2.6}$$

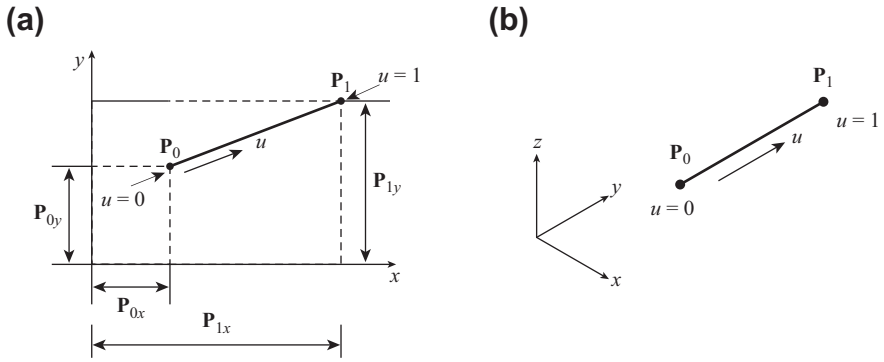


FIGURE 2.4

A straight line defined by its start and end points. (a) Planar. (b) Spatial.

where a_{1x}, a_{1y}, a_{1z}, a_{0x}, a_{0y}, and a_{0z} are unknown coefficients to be determined by the locations of start and end points $\mathbf{P}_0$ and $\mathbf{P}_1$. Rewriting Eq. 2.6 in a matrix form, we have

$$\mathbf{P}(u) = [\,u \quad 1\,]_{1\times 2}\begin{bmatrix} a_{1x} & a_{1y} & a_{1z} \\ a_{0x} & a_{0y} & a_{0z} \end{bmatrix}_{2\times 3} = \mathbf{U}_{1\times 2}\mathbf{A}_{2\times 3} \tag{2.7}$$

where matrix $\mathbf{A}$ contains the unknown coefficients.

By plugging $u = 0$ and $u = 1$ into Eq. 2.7, we have respectively $\mathbf{P}(0) = \mathbf{P}_0 = [0\ 1]\ \mathbf{A}$, and $\mathbf{P}(1) = \mathbf{P}_1 = [1\ 1]\ \mathbf{A}$.

By rewriting the above equations in a matrix form, we have

$$\begin{bmatrix} \mathbf{P}(0) \\ \mathbf{P}(1) \end{bmatrix}_{2\times 3} = \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \end{bmatrix}_{2\times 3} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}_{2\times 2} \mathbf{A}_{2\times 3} \tag{2.8}$$

where $\mathbf{P}_0$ and $\mathbf{P}_1$ must be known in order to define the straight line. From Eq. 2.8, the matrix $\mathbf{A}$ can be obtained by

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{-1}\begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \end{bmatrix}. \tag{2.9}$$

Hence, from Eq. 2.7, we have

$$\mathbf{P}(u) = \mathbf{U}\mathbf{A} = [\,u \quad 1\,]\begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \end{bmatrix} = (1-u)\mathbf{P}_0 + u\mathbf{P}_1 \tag{2.10}$$

where $(1 - u)$ and u are so-called basis functions associated with the characteristic points (in this case, the start and end points $\mathbf{P}_0$ and $\mathbf{P}_1$) of the straight line. The parametric curve equations for other polynomials can be derived in the same way.

2.2.2 QUADRATIC CURVES

A quadratic curve can be created by three distinct points—$\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$, as shown in Figure 2.5(a). Such a curve is called a *spline curve*. In addition to a spline curve, a quadratic curve can be defined by two end points and a vector (Figure 2.5(b)), and by three control points forming a control polygon that encloses a Bézier curve (shown in Figure 2.5(c)), among others.

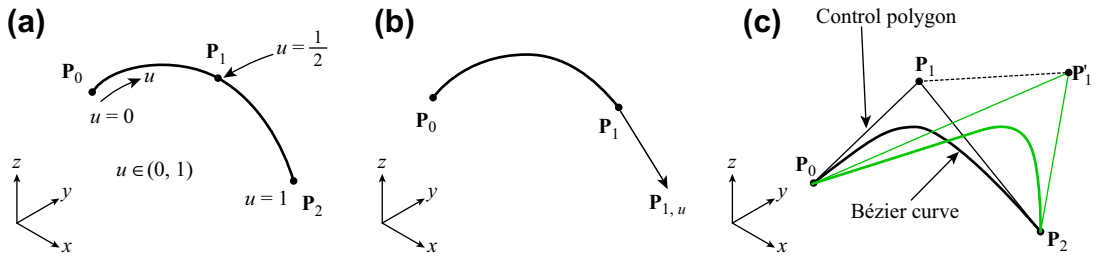


FIGURE 2.5

Quadratic curves defined by (a) three distinct points (spline curve), (b) two end points and a vector, and (c) three control points (Bézier curve).

Similar to Eq. 2.7, a quadratic curve can be written in the following parametric form:

$$\mathbf{P}(u) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix}_{1\times 3} \begin{bmatrix} a_{2x} & a_{2y} & a_{2z} \\ a_{1x} & a_{1y} & a_{1z} \\ a_{0x} & a_{0y} & a_{0z} \end{bmatrix}_{3\times 3} = \mathbf{U}_{1\times 3}\mathbf{A}_{3\times 3} \tag{2.11}$$

where matrix $\mathbf{A}$ contains 9 (3×3) unknown coefficients.

2.2.2.1 Spline Curve—Three Points

For a quadratic spline curve, we assume the three distinct points are $\mathbf{P}_0 = \mathbf{P}(0)$, $\mathbf{P}_1 = \mathbf{P}(½)$, and $\mathbf{P}_2 = \mathbf{P}(1)$. Note that $\mathbf{P}_1$ does not have to be located at $u = ½$.

By plugging $u = 0$, $u = ½$, and $u = 1$ into Eq. 2.11, we have respectively $\mathbf{P}(0) = \mathbf{P}_0 = [0\ 0\ 1]\ \mathbf{A}$, $\mathbf{P}(½) = \mathbf{P}_1 = [¼\ ½\ 1]\ \mathbf{A}$, and $\mathbf{P}(1) = \mathbf{P}_2 = [1\ 1\ 1]\ \mathbf{A}$.

By rewriting the above equations in a matrix form, we have

$$\begin{bmatrix} \mathbf{P}(0) \\ \mathbf{P}\left(\frac{1}{2}\right) \\ \mathbf{P}(1) \end{bmatrix}_{3\times 3} = \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix}_{3\times 3} = \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{4} & \frac{1}{2} & 1 \\ 1 & 1 & 1 \end{bmatrix} \mathbf{A}_{3\times 3} \tag{2.12}$$

where $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$ are known. The matrix $\mathbf{A}$ can be obtained by

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 \\ \frac{1}{4} & \frac{1}{2} & 1 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix} = \begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix}. \tag{2.13}$$

Hence from Eq. 2.11, we have

$$\mathbf{P}(u) = \mathbf{U}\mathbf{A} = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix} = \mathbf{U}\mathbf{N}^s\mathbf{G}^s = \mathbf{B}^s\mathbf{G}^s \tag{2.14}$$

where $\mathbf{N}^s$ is a constant 3×3 matrix for any given quadratic spline curve for which $\mathbf{P}_1$ is at $u = ½$, and $\mathbf{B}^s$ is the 1×3 vector of the basis functions (also called blending functions); that is,

$$\mathbf{B}^{\mathrm{s}}(u) = \mathbf{UN}^{\mathbf{s}} = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix} \tag{2.15}$$

$$= \begin{bmatrix} 2u^2 - 3u + 1, & -4u^2 + 4u, & 2u^2 - \mathrm{u} \end{bmatrix} = \begin{bmatrix} \mathrm{B}_0^{\mathrm{s}}(u), \mathrm{B}_1^{\mathrm{s}}(u), \mathrm{B}_2^{\mathrm{s}}(u) \end{bmatrix}$$

which are plotted in Figure 2.6(a). Note that the superscript s denotes a spline curve.

Therefore, Eq. 2.14 can be rewritten as

$$\mathbf{P}(u) = \mathbf{B}^{\mathrm{s}}\mathbf{G}^{\mathrm{s}} = \begin{bmatrix} 2u^2 - 3u + 1, & -4u^2 + 4u, & 2u^2 - u \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix} \tag{2.16}$$

$$= \mathrm{B}_0^{\mathrm{s}}(u)\mathbf{P}_0 + \mathrm{B}_1^{\mathrm{s}}(u)\mathbf{P}_1 + \mathrm{B}_2^{\mathrm{s}}(u)\mathbf{P}_2.$$

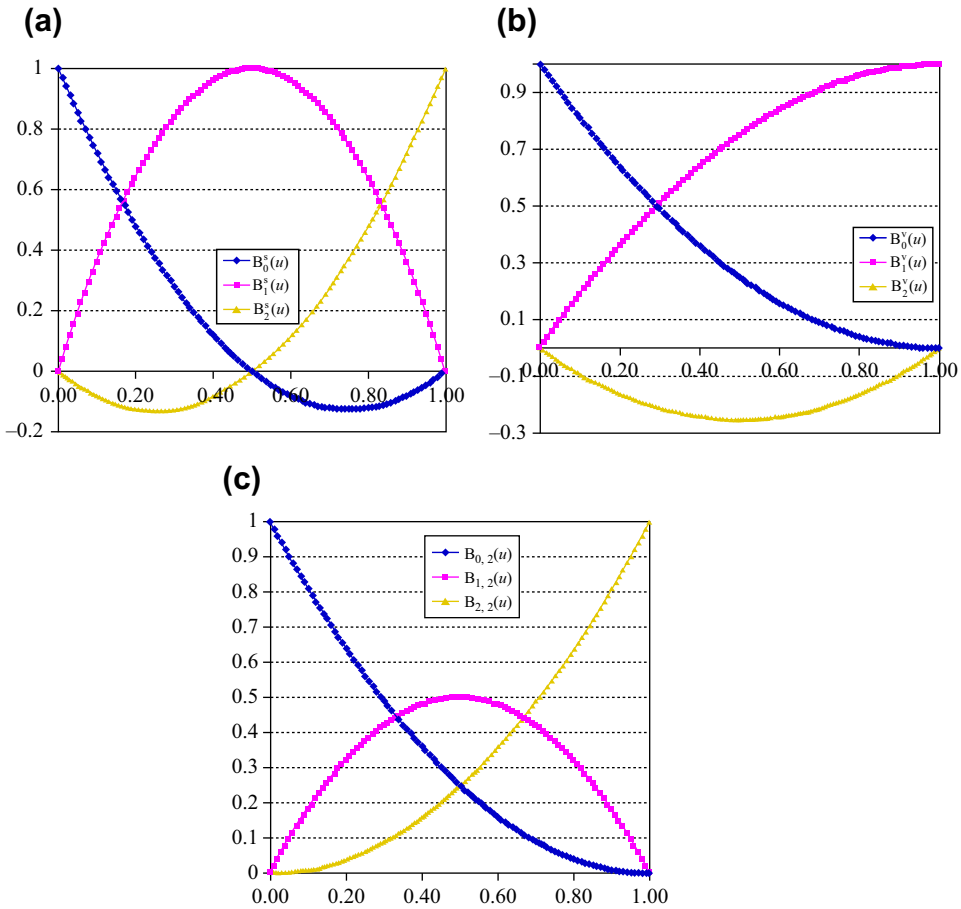


FIGURE 2.6

Basis functions for quadratic curves. (a) Spline curve. (b) Two end points and a vector. (c) Bézier curve.

EXAMPLE 2.1

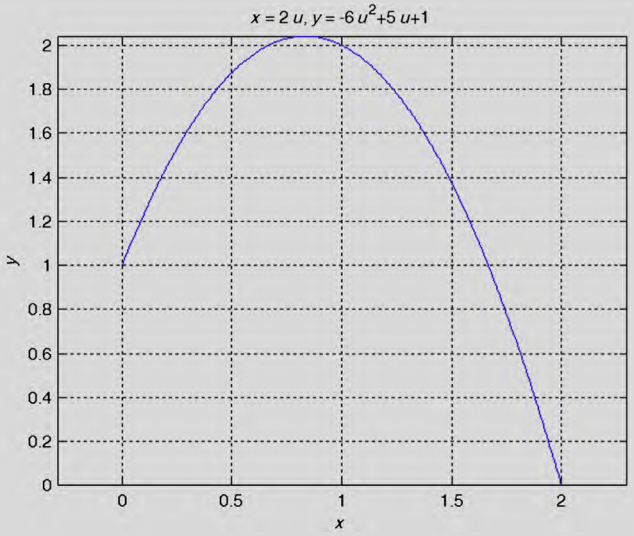
Given three points, $\mathbf{P}_0 = [0,1]$, $\mathbf{P}_1 = [1,2]$, and $\mathbf{P}_2 = [2,0]$, derive the parametric equation and graph the spline curve formed by them.

Solutions

Using Eq. 2.16, we have

$$\mathbf{P}(u) = \mathbf{B}^s\mathbf{G}^s = \begin{bmatrix} 2u^2 - 3u + 1, & -4u^2 + 4u, & 2u^2 - u \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 2u, & -6u^2 + 5u + 1 \end{bmatrix}.$$

The spline curve is graphed in Matlab with script shown below.



Matlab Script:

```
ezplot('2*u','-6*u^2+5*u+1',[0,1])
grid on
```

2.2.2.2 Two Points and a Vector

A quadratic curve can also be defined by its start and end points plus a vector, either at the start or the end point. For example, the curve shown in Figure 2.5(b) is defined by two points $\mathbf{P}_0$ and $\mathbf{P}_1$ at the start and end of the curve and a tangent vector $\mathbf{P}_{1,u}$ at its end point. From Eq. 2.11, the tangent vector $\mathbf{P}_{1,u}$ is defined as

$$\mathbf{P}_{1,u} = \left.\frac{\partial \mathbf{P}(u)}{\partial u}\right|_{u=1} = \left.\frac{\partial (\mathbf{UA})}{\partial u}\right|_{u=1} = \left.\frac{\partial \mathbf{U}}{\partial u}\right|_{u=1} \mathbf{A} = \begin{bmatrix} 2u & 1 & 0 \end{bmatrix}\Big|_{u=1} \mathbf{A} = \begin{bmatrix} 2 & 1 & 0 \end{bmatrix}\mathbf{A} \tag{2.17}$$

Following the same steps as before, we have

$$\begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_{1,u} \end{bmatrix}_{3\times 3} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 0 \end{bmatrix} \mathbf{A}_{3\times 3} \tag{2.18}$$

where $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_{1,u}$ are known. The matrix $\mathbf{A}$ can be obtained by

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_{1,u} \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 \\ -2 & 2 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_{1,u} \end{bmatrix}. \tag{2.19}$$

Hence from Eq. 2.11, we have

$$\mathbf{P}(u) = \mathbf{UA} = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 \\ -2 & 2 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_{1,u} \end{bmatrix} = \mathbf{UN}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} = \mathbf{B}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} \tag{2.20}$$

where $\mathbf{N}^{\mathrm{v}}$ is a constant 3×3 matrix for any given quadratic curve defined by the end points and a tangent vector at the end, and $\mathbf{B}^{\mathrm{v}}$ is the 1×3 vector of the basis functions; that is,

$$\begin{aligned} \mathbf{B}^{\mathrm{v}}(u) = \mathbf{UN}^{\mathrm{v}} &= \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 1 \\ -2 & 2 & -1 \\ 1 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} u^2 - 2u + 1, & -u^2 + 2u, & u^2 - u \end{bmatrix} = \begin{bmatrix} \mathrm{B}_0^{\mathrm{v}}(u), & \mathrm{B}_1^{\mathrm{v}}(u), & \mathrm{B}_2^{\mathrm{v}}(u) \end{bmatrix} \end{aligned} \tag{2.21}$$

which are plotted in Figure 2.6(b). Note that the superscript v denotes a curve with a tangent vector. Therefore, from Eq. 2.20, we have

$$\begin{aligned} \mathbf{P}(u) = \mathbf{B}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} &= \begin{bmatrix} u^2 - 2u + 1, & -u^2 + 2u, & u^2 - u \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix} \\ &= \mathrm{B}_0^{\mathrm{v}}(u)\mathrm{P}_0 + \mathrm{B}_1^{\mathrm{v}}(u)\mathrm{P}_1 + \mathrm{B}_2^{\mathrm{v}}(u)\mathrm{P}_{1,u}. \end{aligned} \tag{2.22}$$

Note that the geometric shape of the curve is controlled by the tangent vector in addition to its start and end points. The basis function associated with the tangent vector $\mathrm{B}_2^{\mathrm{v}}(u)$ is negative for $u \in [0,1]$, indicating that when the vector size increases, the curve is "pushed" backward, as shown in Example 2.2.

EXAMPLE 2.2

Given two points and a tangent vector at the end point, $\mathbf{P}_0 = [0,1]$, $\mathbf{P}_1 = [2,0]$, and $\mathbf{P}_{1,u} = [2,-7]$, derive the parametric equation and graph the curve formed by them. Also, graph the curve by changing the tangent vector to $\mathbf{P}_{1,u} = [3,-10.5]$.

Solutions

Using Eq. 2.22, we have

$$\begin{aligned} \mathbf{P}(u) = \mathbf{B}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} &= \begin{bmatrix} u^2 - 2u + 1, & -u^2 + 2u, & u^2 - u \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 0 \\ 2 & -7 \end{bmatrix} \\ &= \begin{bmatrix} 2u, & -6u^2 + 5u + 1 \end{bmatrix}. \end{aligned}$$

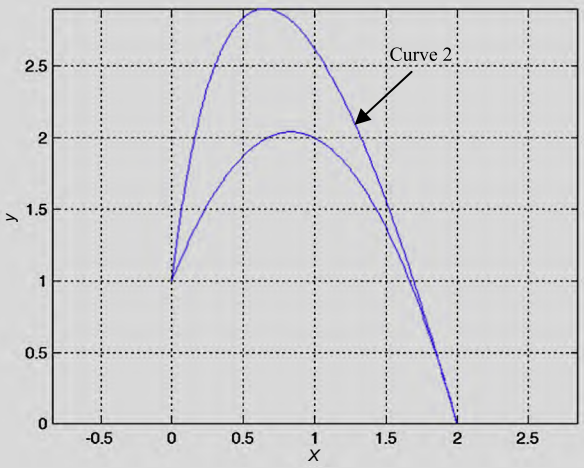
Continued

EXAMPLE 2.2—cont'd

Using the tangent vector of a larger size, the curve (Curve 2) becomes

$$\mathbf{P}(u) = \mathbf{B}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} = \begin{bmatrix} u^2 - 2u + 1, & -u^2 + 2u, & u^2 - u \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 2 & 0 \\ 3 & -10.5 \end{bmatrix}$$
$$= \begin{bmatrix} u^2 + u, & -9.5u^2 + 8.5u + 1 \end{bmatrix}.$$

The curves are graphed in Matlab with script shown below. It is clearly shown in the figure that when the vector size increases in Curve 2, the curve is "pushed" backward (in this case, upward and to the left).



Matlab Script:

```
ezplot('2*u','-6*u^2+5*u+1',[0,1])
grid on
hold on
ezplot('u^2+u','-9.5*u^2+8.5*u+1',[0,1])
```

2.2.2.3 Bézier Curve

The Bézier curve, originally developed by Pierre Bézier in the 1970s, has become one of the most commonly used curves for geometric modeling. As shown in Figure 2.6(c), unlike a spline curve, a Bézier curve is defined by control points that do not necessarily stay on the curve. The control points form a control polygon (or characteristic polygon) that determines the shape of the curve. More specifically, in general, only the first and last control points stay on the curve; in fact, in this case they coincide with the start and end points of the curve, respectively. The curve is also tangent to the first and last line segments of the control polygon, which provides the designer with direct control of the geometric shape of the curve at the ends. In addition to controlling the tangent vectors of the curves at ends, changing the control point locations alters the shape of the curve, as illustrated in Figure 2.5(c), in which the control point $\mathbf{P}_1$ is moved to a new location $\mathbf{P}_1'$. Figure 2.7 also illustrates this point, in which control point $\mathbf{P}_2'$ is relocated.

Mathematically, a Bézier curve is defined as

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i \mathrm{B}_{i,n}(u), \quad u \in [0,1] \tag{2.23}$$

where $\mathbf{P}_i$ is the ith control point, n is the polynomial order of the curve, and $\mathrm{B}_{i,n}(u)$ is the corresponding basis function, called the Bernstein polynomial, defined as

$$\mathrm{B}_{i,n}(u) = \mathrm{C}(n,i)u^i(1-u)^{n-i}, \quad u \in [0,1]. \tag{2.24}$$

$\mathrm{C}(n,i)$ is the binomial coefficient defined as

$$\mathrm{C}(n,i) = \frac{n!}{i!(n-i)!} = \binom{n}{i}. \tag{2.25}$$

Note that we assume $u^0 = 1$, including when u = 0, and 0! = 1, in Eqs 2.24 and 2.25, respectively.

For a quadratic Bézier curve, $n = 2$, and the curve is defined by three control points. From Eq. 2.24, the basic functions of a quadratic curve can be derived as follows:

$$\begin{aligned} \mathrm{B}_{0,2}(u) &= (1-u)^2 \\ \mathrm{B}_{1,2}(u) &= 2u(1-u) \\ \mathrm{B}_{2,2}(u) &= u^2 \end{aligned} \tag{2.26}$$

which are plotted in Figure 2.6(c). Note that the sum of all three basis functions is 1—that is, $\sum_{i=0}^{2} \mathrm{B}_{i,2}(u) = 1$, implying that the basis functions form a partition of unity. The partition of unity is a very important property when utilizing Bernstein polynomials in geometric modeling. In particular, for

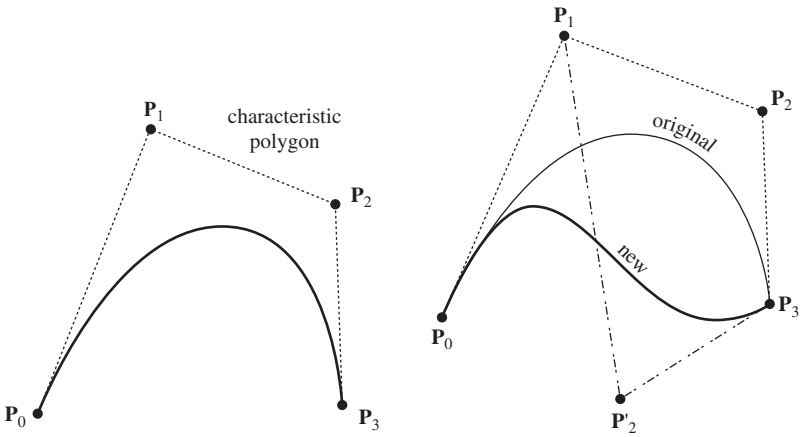


FIGURE 2.7

The geometric shape of a cubic Bézier curve as determined by its control polygon.

any set of control points $\mathbf{P}_0$, $\mathbf{P}_1$,…, $\mathbf{P}_n$, in a two- or three-dimensional space and for any u in [0,1], the expression of Eq. 2.23 is a convex combination of the set of points $\mathbf{P}_0$, $\mathbf{P}_1$,…, $\mathbf{P}_n$. Note that a convex combination is a linear combination of vectors (e.g., in this case, control points in Bézier curve), where all coefficients are nonnegative and sum up to 1. In geometric modeling, a curve of convex combination implies the convex hull (or convex envelop) property, which ensures that the curve lies within the convex hull of the control points (that is, the control polygon).

It can be shown for a given n that the Bernstein polynomials $\mathrm{B}_{i,n}(u)$ satisfy the following:

1. $0 \leq \mathrm{B}_{i,n}(u) \leq 1$ for $u \in [0,1]$, and
2. $\sum_{i=0}^{2} \mathrm{B}_{i,n}(u) = 1$

It is apparent that a Bézier curve, written in Eq. 2.23, is a convex combination of the control points; therefore the curve lies within the convex hull of the control points.

A quadratic Bézier curve shown in Eq. 2.23 can be explicitly written as

$$\begin{aligned}\mathbf{P}(\mathrm{u}) &= \sum_{i=0}^{2} \mathbf{P}_i \mathrm{B}_{i,2}(\mathrm{u}) = \mathbf{P}_0 \mathrm{B}_{0,2}(\mathrm{u}) + \mathbf{P}_1 \mathrm{B}_{1,2}(\mathrm{u}) + \mathbf{P}_2 \mathrm{B}_{2,2}(\mathrm{u}) \\ &= \mathbf{P}_0(1-\mathrm{u})^2 + \mathbf{P}_1(2\mathrm{u}(1-\mathrm{u})) + \mathbf{P}_2\mathrm{u}^2, \quad \mathrm{u} \in [0,1].\end{aligned} \tag{2.27}$$

Note that Eq. 2.27 can also be written in a matrix form, similar to those of Eqs 2.14 and 2.20, as

$$\mathbf{P}(u) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix} = \mathbf{U}\mathbf{N}^{\mathrm{B}}\mathbf{G}^{\mathrm{B}} = \mathbf{B}^{\mathrm{B}}\mathbf{G}^{\mathrm{B}} \tag{2.28}$$

where $\mathbf{N}^{\mathrm{B}}$ is a constant 3×3 matrix for any given quadratic Bézier curve, and $\mathbf{B}^{\mathrm{B}}$ is the 1×3 vector of the basis functions; in this case, they are Bernstein polynomials.

EXAMPLE 2.3

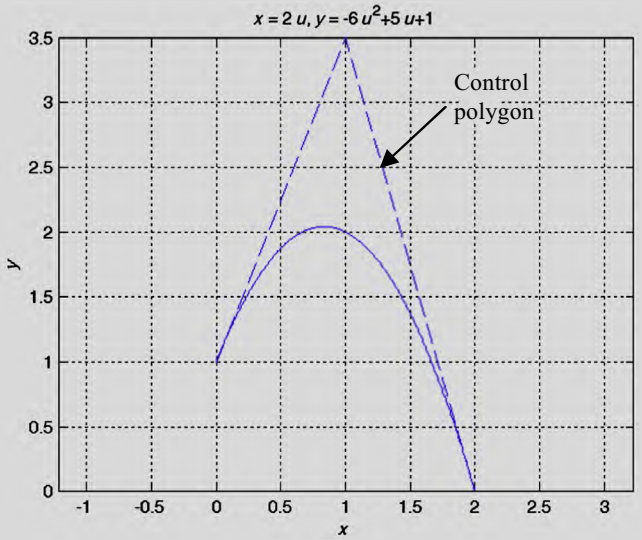
Given three control points, $\mathbf{P}_0 = [0,1]$, $\mathbf{P}_1 = [1,3.5]$, and $\mathbf{P}_2 = [2,0]$, derive the parametric equation and graph the Bézier curve formed by them.

Solutions

Using Eq. 2.28, we have

$$\begin{aligned}\mathbf{P}(u) = \mathbf{B}^{\mathrm{B}}\mathbf{G}^{\mathrm{B}} &= \begin{bmatrix} (1-u)^2, & 2u(1-u), & u^2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 3.5 \\ 2 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 2u, & -6u^2 + 5u + 1 \end{bmatrix}.\end{aligned}$$

EXAMPLE 2.3—cont'd



Matlab Script:

```
ezplot('2*u','-6*u^2+5*u+1',[0,1])
grid on
hold on
Px=[0 1 2]
Py=[1 3.5 0]
Plot(Px,Py,'--')
```

As shown in Examples 2.1–2.3, all three curves represented in their respective forms are identical, which implies that the same curve can be represented in different forms. This is because all forms are representing a quadratic curve, which is a second-order polynomial function. Mathematically, they are all identical; therefore, we have the following:

$$\mathbf{P}(u) = \mathbf{UA} = \mathbf{UN}^{\mathrm{s}}\mathbf{G}^{\mathrm{s}} = \mathbf{UN}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} = \mathbf{UN}^{\mathrm{B}}\mathbf{G}^{\mathrm{B}}. \tag{2.29}$$

Eq. 2.29 implies that curves can be transformed into various forms to meet different modeling requirements. For example, a quadratic spline curve can be converted to a two point and a vector form, as well as a Bézier curve as, respectively,

$$\mathbf{G}^{\mathrm{v}} = \mathbf{N}^{\mathrm{v}^{-1}}\mathbf{N}^{\mathrm{s}}\mathbf{G}^{\mathrm{s}} \tag{2.30a}$$

and

$$\mathbf{G}^{\mathrm{B}} = \mathbf{N}^{\mathrm{B}^{-1}}\mathbf{N}^{\mathrm{s}}\mathbf{G}^{\mathrm{s}} \tag{2.30b}$$

The same is true for converting a two point and a vector or a Bézier curve to other forms.

EXAMPLE 2.4

Given a spline curve formed by three points, $\mathbf{P}_0 = [0,1]$, $\mathbf{P}_1 = [1,2]$, and $\mathbf{P}_2 = [2,0]$, convert the curve into two points with a vector form and then a Bézier curve.

Continued

EXAMPLE 2.4—cont'd

Solutions

Using Eqs 2.30a and 2.30b, we have

$$\mathbf{G}^{\mathrm{v}} = \mathbf{N}^{\mathrm{v}^{-1}}\mathbf{N}^{\mathrm{s}}\mathbf{G}^{\mathrm{s}} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 2 & 0 \\ 2 & -7 \end{bmatrix}$$

in which $\mathbf{G}^{\mathrm{v}}$ contains the end points and tangent vector of the quadratic curve, and

$$\mathbf{G}^{\mathrm{B}} = \mathbf{N}^{\mathrm{B}^{-1}}\mathbf{N}^{\mathrm{s}}\mathbf{G}^{\mathrm{s}} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 3.5 \\ 2 & 0 \end{bmatrix}$$

in which $\mathbf{G}^{\mathrm{B}}$ contains the three control points of the quadratic Bézier curve.

2.2.3 CUBIC CURVES

Similar to Eq. 2.11, a cubic curve can be written in the following parametric form:

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}_{1\times 4} \begin{bmatrix} a_{3x} & a_{3y} & a_{3z} \\ a_{2x} & a_{2y} & a_{2z} \\ a_{1x} & a_{1y} & a_{1z} \\ a_{0x} & a_{0y} & a_{0z} \end{bmatrix}_{4\times 3} = \mathbf{U}_{1\times 4}\mathbf{A}_{4\times 3} \tag{2.31}$$

where matrix $\mathbf{A}$ contains 12 (4×3) unknown coefficients.

2.2.3.1 *Spline Curve—Four Points*

For a cubic spline curve, we assume the four distinct points are $\mathbf{P}_0 = \mathbf{P}(0)$, $\mathbf{P}_1 = \mathbf{P}(1/3)$, $\mathbf{P}_2 = \mathbf{P}(2/3)$, and $\mathbf{P}_3 = \mathbf{P}(1)$, as shown in Figure 2.8(a). Note that $\mathbf{P}_1$ and $\mathbf{P}_2$ can be at locations other than $u = 1/3$ or 2/3.

Following the same idea as the quadratic curves, we plug $u = 0$, $u = 1/3$, $u = 2/3$, and $u = 1$ into Eq. 2.31 to yield

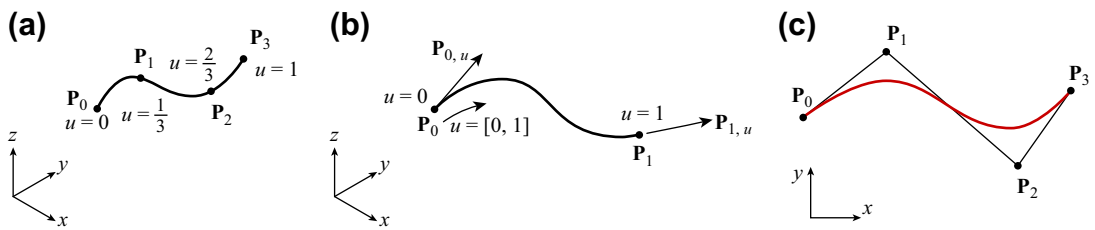


FIGURE 2.8

Cubic curves defined by (a) four distinct points (spline curve), (b) end points and end vectors (Hermit cubic curve), and (c) four control points (Bézier curve).

$$\begin{bmatrix} \mathbf{P}(0) \\ \mathbf{P}(1/3) \\ \mathbf{P}(2/3) \\ \mathbf{P}(1) \end{bmatrix}_{4\times 3} = \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}_{4\times 3} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1/27 & 1/9 & 1/3 & 1 \\ 8/27 & 4/9 & 2/3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \mathbf{A}_{4\times 3} \tag{2.32}$$

where $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ must be known in order to define the curve. The matrix $\mathbf{A}$ can be obtained by

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1/27 & 1/9 & 1/3 & 1 \\ 8/27 & 4/9 & 2/3 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} = \begin{bmatrix} -9/2 & 27/2 & -27/2 & 9/2 \\ 9 & -45/2 & 18 & -9/2 \\ -11/2 & 9 & -9/2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} \tag{2.33}$$

Hence, from Eq. 2.31, we have

$$\mathbf{P}(u) = \mathbf{UA} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -9/2 & 27/2 & -27/2 & 9/2 \\ 9 & -45/2 & 18 & -9/2 \\ -11/2 & 9 & -9/2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} = \mathbf{UN}^{\mathrm{s}}\mathbf{G}^{\mathrm{s}} = \mathbf{B}^{\mathrm{s}}\mathbf{G}^{\mathrm{s}} \tag{2.34}$$

where $\mathbf{N}^{\mathrm{s}}$ is a constant 4×4 matrix for any given cubic spline curve, and $\mathbf{B}^{\mathrm{s}}$ is the 1×4 vector of the basis functions (also called blending functions); that is,

$$\begin{aligned} \mathbf{B}^{\mathrm{s}}(u) = \mathbf{UN}^{\mathrm{s}} &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -9/2 & 27/2 & -27/2 & 9/2 \\ 9 & -45/2 & 18 & -9/2 \\ -11/2 & 9 & -9/2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} -9/2u^3 + 9u^2 - 11/2u + 1, & 27/2u^3 - 45/2u^2 + 9u, \\ -27/2u^3 + 18u^2 - 9/2u + 1, & 9/2u^3 + 9/2u^2 + u \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{B}_0^{\mathrm{s}}(u), & \mathrm{B}_1^{\mathrm{s}}(u), & \mathrm{B}_2^{\mathrm{s}}(u), & \mathrm{B}_3^{\mathrm{s}}(u) \end{bmatrix} \end{aligned} \tag{2.35}$$

which are plotted in Figure 2.9(a). Therefore, from Eq. 2.34, we have

$$\begin{aligned} \mathbf{P}(u) = \mathbf{B}^{\mathrm{s}}\mathbf{G}^{\mathrm{s}} &= \begin{bmatrix} \mathrm{B}_0^{\mathrm{s}}(u), & \mathrm{B}_1^{\mathrm{s}}(u), & \mathrm{B}_2^{\mathrm{s}}(u), & \mathrm{B}_3^{\mathrm{s}}(u) \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} \\ &= \mathrm{B}_0^{\mathrm{s}}(u)\mathbf{P}_0 + \mathrm{B}_1^{\mathrm{s}}(u)\mathbf{P}_1 + \mathrm{B}_2^{\mathrm{s}}(u)\mathbf{P}_2 + \mathrm{B}_3^{\mathrm{s}}(u)\mathbf{P}_3. \end{aligned} \tag{2.36}$$

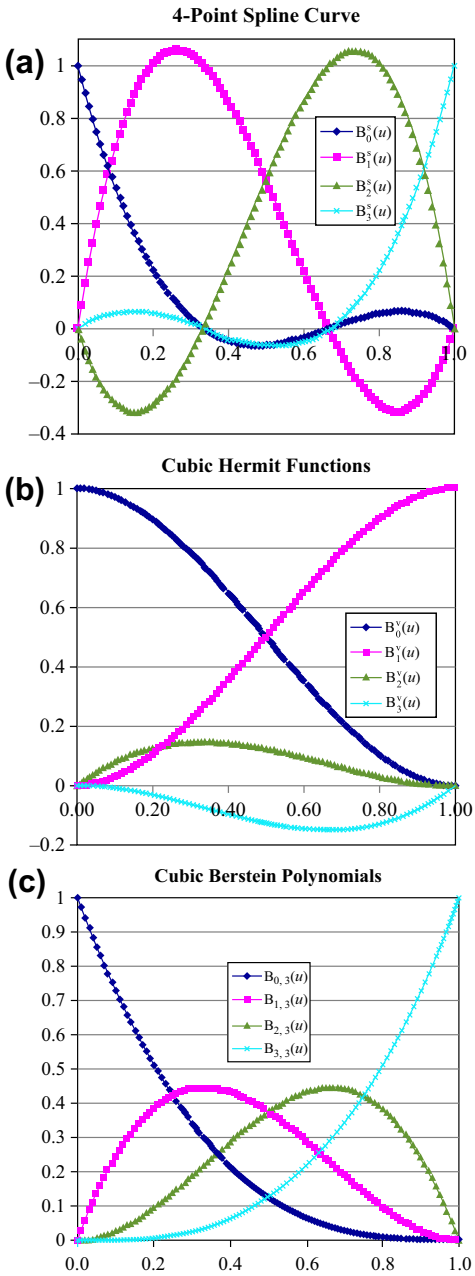


FIGURE 2.9

Basis functions for cubic parametric curves. (a) Four-point spline curve. (b) Hermit cubic curve. (c) Cubic Bernstein polynomials.

EXAMPLE 2.5

Given four points, $\mathbf{P}_0 = [0,1]$, $\mathbf{P}_1 = [1,2]$, $\mathbf{P}_2 = [2,0]$, and $\mathbf{P}_3 = [4,-1]$, derive the parametric equation and graph the spline curve formed by them.

Solutions

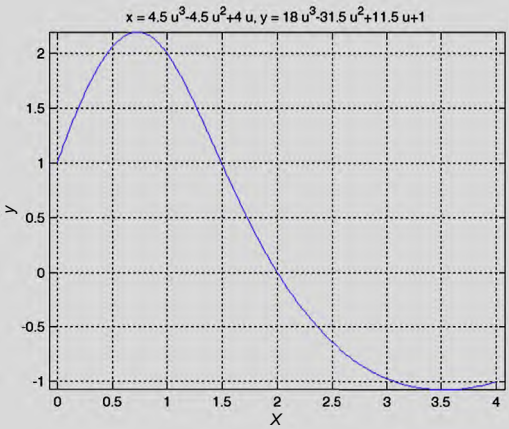
Using Eq. 2.36, we have

$$\mathbf{P}(u) = \mathbf{B}^s\mathbf{G}^s = [-9/2u^3 + 9u^2 - 11/2u + 1, 27/2u^3 - 45/2u^2 + 9u,$$

$$-27/2u^3 + 18u^2 - 9/2u + 1, 9/2u^3 + 9/2u^2 + u] \begin{bmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 0 \\ 4 & -1 \end{bmatrix}$$

$$= [9/2u^3 - 9/2u^2 + 4u, 18u^3 - 63/2u^2 + 23/2u + 1].$$

The cubic spline curve is graphed in Matlab with script shown below.



Matlab Script:

```
ezplot('4.5*u^3-4.5*u^2+4*u','18*u^3-31.5*u^2+11.5*u+1',
[0,1])
grid on
```

2.2.3.2 Hermit Cubic Curve (Two End Points and Two End Vectors)

Similar to the quadratic curve, a cubic curve can also be defined by its start and end points plus tangent vectors at the start and end points, as shown in Figure 2.8(b), in which the tangent vectors $\mathbf{P}_{0,u}$ and $\mathbf{P}_{1,u}$ are at the start and end points, respectively. This is called a Hermit cubic curve or curve of geometric format. Note that from Eq. 2.34, the tangent vectors $\mathbf{P}_{0,u}$ and $\mathbf{P}_{1,u}$ are defined as

$$\mathbf{P}_{0,u} = \frac{\partial \mathbf{P}(u)}{\partial u}\Big|_{u=0} = \frac{\partial(\mathbf{UA})}{\partial u}\Big|_{u=0} = \frac{\partial \mathbf{U}}{\partial u}\Big|_{u=0}\mathbf{A} = [3u^2 \quad 2u \quad 1 \quad 0]\big|_{u=0}\mathbf{A} = [0 \quad 0 \quad 1 \quad 0]\mathbf{A}$$

$$\mathbf{P}_{1,u} = \frac{\partial \mathbf{P}(u)}{\partial u}\Big|_{u=1} = \frac{\partial(\mathbf{UA})}{\partial u}\Big|_{u=1} = \frac{\partial \mathbf{U}}{\partial u}\Big|_{u=1}\mathbf{A} = [3u^2 \quad 2u \quad 1 \quad 0]\big|_{u=1}\mathbf{A} = [3 \quad 2 \quad 1 \quad 0]\mathbf{A}. \tag{2.37}$$

Following the same steps as before, we have

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_{0,u} \\ \mathbf{P}_{1,u} \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_{0,u} \\ \mathbf{P}_{1,u} \end{bmatrix}. \tag{2.38}$$

Hence from Eq. 2.31, we have

$$\mathbf{P}(\mathrm{u}) = \mathbf{UA} = \begin{bmatrix} \mathrm{u}^3 & \mathrm{u}^2 & \mathrm{u} & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_{0,\mathrm{u}} \\ \mathbf{P}_{1,\mathrm{u}} \end{bmatrix} = \mathbf{UN}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} = \mathbf{B}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} \tag{2.39}$$

where $\mathbf{N}^{\mathrm{v}}$ is a constant 4×4 matrix for any given cubic curve defined by the end points and vectors, and $\mathbf{B}^{\mathrm{v}}$ is the 1×4 vector of the basis functions called cubic Hermit functions; that is,

$$\begin{aligned} \mathbf{B}^{\mathrm{v}}(u) = \mathbf{UN}^{\mathrm{v}} &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 2u^3 - 3u^2 + 1, & -2u^3 + 3u^2 + 1, & u^3 - 2u^2 + u, & u^3 - u^2 \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{B}_0^{\mathrm{v}}(u), & \mathrm{B}_1^{\mathrm{v}}(u), & \mathrm{B}_2^{\mathrm{v}}(u), & \mathrm{B}_3^{\mathrm{v}}(u) \end{bmatrix} \end{aligned} \tag{2.40}$$

which are plotted in Figure 2.9(b). Therefore, from Eq. 2.39, we have

$$\begin{aligned} \mathbf{P}(u) = \mathbf{B}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} &= \begin{bmatrix} 2u^3 - 3u^2 + 1, & -2u^3 + 3u^2 + 1, & u^3 - 2u^2 + u, & u^3 - u^2 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_{0,\mathrm{u}} \\ \mathbf{P}_{1,\mathrm{u}} \end{bmatrix} \\ &= \mathrm{B}_0^{\mathrm{v}}(u)\mathbf{P}_0 + \mathrm{B}_1^{\mathrm{v}}(u)\mathbf{P}_1 + \mathrm{B}_2^{\mathrm{v}}(u)\mathbf{P}_{0,u} + \mathrm{B}_3^{\mathrm{v}}(u)\mathbf{P}_{1,u}. \end{aligned} \tag{2.41}$$

Similar to the quadratic curve, the geometric shape of the Hermit cubic curve is controlled by the tangent vectors in addition to its start and end points. The basis function $\mathrm{B}_2^{\mathrm{v}}(u)$, which is associated with the tangent vector at the start point of the curve, is positive for $\mathrm{u} \in [0,1]$, indicating that when the size of the tangent vector $\mathbf{P}_{0,u}$ increases, the curve is "pulled" forward. On the other hand, the basis function $\mathrm{B}_3^{\mathrm{v}}(u)$ is negative for $u \in [0,1]$, indicating that when the size of the vector $\mathbf{P}_{1,u}$ increases, the curve is "pushed" backward similar to that of the quadratic curve.

EXAMPLE 2.6

Given two points and two vectors, $\mathbf{P}_0 = [0,0]$, $\mathbf{P}_1 = [1,1]$, $\mathbf{P}_{0,u} = [2,0]$, and $\mathbf{P}_{1,u} = [2,0]$, derive the parametric equation and graph the Hermit cubic curve formed by them. Graph the curve by changing the tangent vector from $\mathbf{P}_{0,u} = [2,0]$ to $\mathbf{P}_{0,u} = [4,0]$. Restore the original curve shape, and then change $\mathbf{P}_{1,u} = [2,0]$ to $\mathbf{P}_{1,u} = [4,0]$.

EXAMPLE 2.6—cont'd

Solutions

Using Eq. 2.41, we have Curve 1:

$$\mathbf{P}(u) = \mathbf{B}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} = \begin{bmatrix} 2u^3 - 3u^2 + 1, & -2u^3 + 3u^2 + 1, & u^3 - 2u^2 + u, & u^3 - u^2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 0 \\ 2 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2u^3 - 3u^2 + 2u, & -2u^3 + 3u^2 \end{bmatrix}.$$

Changing the tangent vector $\mathbf{P}_{0,u} = [2,0]$ to $\mathbf{P}_{0,u} = [4,0]$, the curve becomes Curve 2:

$$\mathbf{P}(u) = \mathbf{B}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} = \begin{bmatrix} 2u^3 - 3u^2 + 1, & -2u^3 + 3u^2 + 1, & u^3 - 2u^2 + u, & u^3 - u^2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 4 & 0 \\ 2 & 0 \end{bmatrix}$$

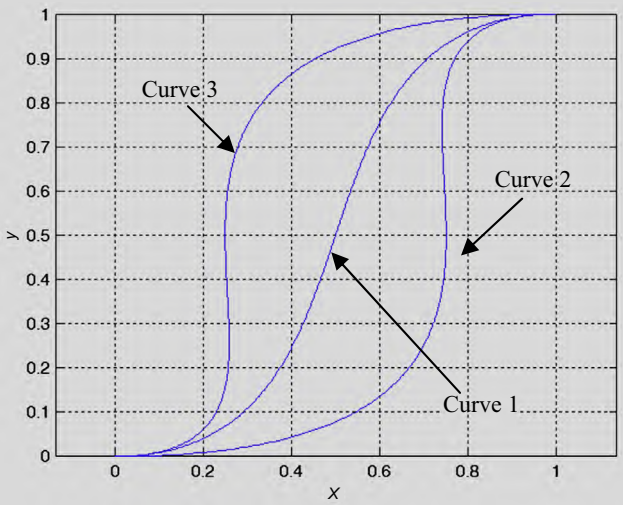
$$= \begin{bmatrix} 4u^3 - 7u^2 + 4u, & -2u^3 + 3u^2 \end{bmatrix}.$$

Go back to Curve 1 and change the tangent vector $\mathbf{P}_{1,u} = [2,0]$ to $\mathbf{P}_{1,u} = [4,0]$. The curve becomes Curve 3:

$$\mathbf{P}(u) = \mathbf{B}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}} = \begin{bmatrix} 2u^3 - 3u^2 + 1, & -2u^3 + 3u^2 + 1, & u^3 - 2u^2 + u & u^3 - u^2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 0 \\ 4 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 4u^3 - 5u^2 + 2\mathrm{u}, & -2u^3 + 3u^2 \end{bmatrix}.$$

The curves are graphed in Matlab with the script shown below. Curve 2 clearly shows that when the vector size at the start point of the curve increases, the curve is "pulled" forward. On the other hand, Curve 3 shows that when the vector size increases at the end point of the curve, the curve is "pushed" backward.



Matlab Script:

```
ezplot('2*u^3-3*u^2+2*u','-
2*u^3+3*u^2',[0,1])
grid on
hold on
ezplot('4*u^3-7*u^2+4*u','-
2*u^3+3*u^2',[0,1])
ezplot('4*u^3-5*u^2+2*u','-
2*u^3+3*u^2',[0,1])
```

2.2.3.3 Bézier Curve

Following the discussion of the quadratic Bézier curve, a cubic Bézier curve consists of four control points. It can be derived as

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} = \mathbf{U}\mathbf{N}^{\mathrm{B}}\mathbf{G}^{\mathrm{B}} = \mathbf{B}^{\mathrm{B}}\mathbf{G}^{\mathrm{B}} \tag{2.42}$$

where $\mathbf{N}^{\mathrm{B}}$ is a constant 4×4 matrix for any given cubic Bézier curve, and $\mathbf{B}^{\mathrm{B}} = [\mathrm{B}_{0,3}(u), \mathrm{B}_{1,3}(u), \mathrm{B}_{2,3}(u), \mathrm{B}_{3,3}(u)]$ is the 1×4 vector of the basis functions (Bernstein polynomials), as plotted in Figure 2.9(c). Derivation of the basis functions is left as an exercise. One interesting point is that when a control point is added to the same location as the existing one, the Bézier curve gets closer to the control polygon, as illustrated in Figure 2.10.

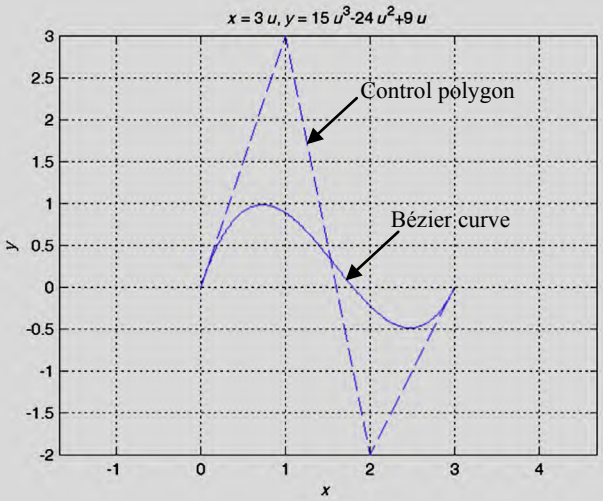
EXAMPLE 2.7

Given four control points, $\mathbf{P}_0 = [0,0]$, $\mathbf{P}_1 = [1,3]$, $\mathbf{P}_2 = [2,-2]$, and $\mathbf{P}_3 = [3,0]$, compute the parametric equation and graph the Bézier curve formed by them.

Solutions

Using Eq. 2.42, we have

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 3 \\ 2 & -2 \\ 3 & 0 \end{bmatrix} = \begin{bmatrix} 3u, & 15u^3 - 24u^2 + 9u \end{bmatrix}.$$



MATLAB Script:

```
ezplot('3*u','15*u^3-24*u^2+9*u',[0,1])
grid on
hold on
Px=[0 1 2 3]
Py=[0 3 -2 0]
plot(Px,Py,'--')
```

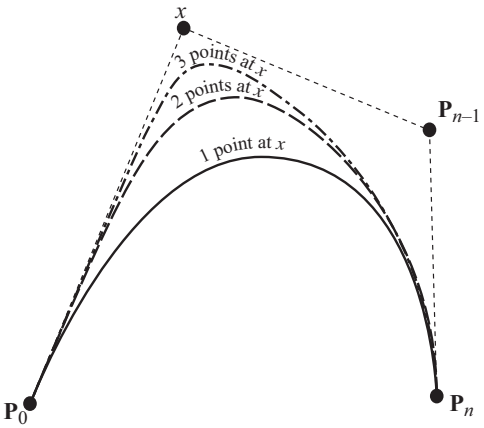


FIGURE 2.10

Increasing the "pull" with coincident points at *x*.

2.2.4 CONTINUITIES

In geometric modeling, curves are joined with smooth transitions at the junction. As illustrated in Figure 2.11, when joining two curves, the end points must coincide; that is, $\mathbf{P}_1^{\mathrm{A}} = \mathbf{P}_0^{\mathrm{B}}$, in which $\mathbf{P}_1^{\mathrm{A}}$ is the end point of curve A and $\mathbf{P}_0^{\mathrm{B}}$ is the start point of curve B. This is so-called C^0-continuity. In order to maintain smoothness, the slope of the curves must be continuous across the junctions. For example, the tangent vectors of Hermit cubic curves shown in Figure 2.11(a) must be collinear (G^1-continuity) or identical (C^1-continuity) at the junction; that is, $\mathbf{P}_{1,u}^{\mathrm{A}} = \mathrm{C}\mathbf{P}_{0,u}^{\mathrm{B}}$ ($\mathrm{C} \neq 0$) or $\mathbf{P}_1^{\mathrm{A}} = \mathbf{P}_0^{\mathrm{B}}$, respectively. Also, for the Bézier curves shown in Figure 2.11(b), the line segments of the respective control polygons must be either collinear (G^1-continuity) or identical (C^1-continuity); that is, $\mathbf{P}_2^{\mathrm{A}}\mathbf{P}_3^{\mathrm{A}} = \mathrm{C}\mathbf{P}_0^{\mathrm{B}}\mathbf{P}_1^{\mathrm{B}}$ ($\mathrm{C} \neq 0$) or $\mathbf{P}_2^{\mathrm{A}}\mathbf{P}_3^{\mathrm{A}} = \mathbf{P}_0^{\mathrm{B}}\mathbf{P}_1^{\mathrm{B}}$, respectively.

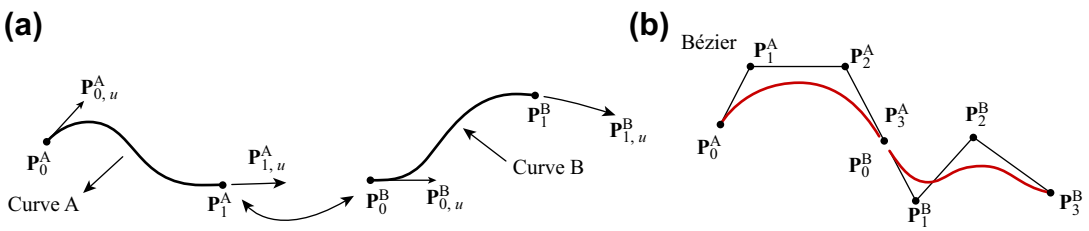


FIGURE 2.11

Curve continuity. (a) Joining two Hermit cubic curves. (b) Joining two Bézier curves.

2.2.5 B-SPLINE CURVES

In general, the parametric curves discussed so far reveal several important characteristics. First, the polynomial order of the curve increases as more points are added. For example, three points form a quadratic spline curve, and four points form a cubic spline curve. Similarly, the polynomial order of a Bézier curve is determined by the total number of control points minus one. The issue with high-order polynomial curves is that as the polynomial order increases, there is a possibility that the curve may oscillate, which is undesirable for geometric modeling. Another characteristic of the curves we discussed so far is that when a point (or a tangent vector) is moved, the entire curve is affected. This is so-called global control, which may not be ideal when a designer only intends to alter the geometric shape of a part in a local area. Moreover, as shown in Section 2.2.4, conditions must be imposed at the curve junctions in order to maintain the desired geometric smoothness.

A better alternative that alleviates the above less-desirable characteristics is the B-spline curve. The power of B-spline curves is that the designer can create with ease a very complex curve that is smoothly connected. The number of control points and the polynomial order are defined separately. In other words, adding a control point does not increase the polynomial order of the curve. In addition, when a control point is relocated, only a portion of the curve is affected, which is referred to as local control. One most important characteristics of the B-spline curve is that the entire curve is smooth. The derivative up to the $(p-1)^{\text{th}}$ order is continuous, where p is the polynomial order of the curve. That is, for a cubic B-spline curve, its curvature (involving a second-order derivative) is continuous throughout the curve.

2.2.5.1 Nonuniform B-Spline Curves

A B-spline curve, more specifically nonuniform B-spline curve, can be defined mathematically as

$$\mathbf{P}(u) = \sum_{i=0}^{n} \mathbf{P}_i \mathrm{N}_{i,k}(u), u \in [0, (n+1)-(k-1)] \tag{2.43}$$

where $n+1$ is the number of control points, $p = k-1$ is the polynomial order, and $\mathrm{N}_{i,k}(u)$'s are the basis functions, which are defined recursively as

$$\mathrm{N}_{i,k}(u) = \frac{(u-t_i)\mathrm{N}_{i,k-1}(u)}{t_{i+k-1}-t_i} + \frac{(t_{i+k}-u)\mathrm{N}_{i+1,k-1}(u)}{t_{i+k}-t_{i+1}} \tag{2.44}$$

and

$$\mathrm{N}_{i,1}(u) = \begin{cases} 1, & t_i \le u \le t_{i+1} \\ 0, & \text{elsewhere.} \end{cases} \tag{2.45}$$

Note that t is called knots in Eqs 2.44 and 2.45, which is defined as

$$\mathrm{t_i} = \begin{cases} 0, & i < k \\ i-k+1, & k \le i \le n. \\ n-k+2, & i > n \end{cases} \tag{2.46}$$

There are $n+k+1$ knots. Note that we defined $0/0 \equiv 0$ in Eq. 2.44.

We will use a quadratic B-spline curve example shown in Figure 2.12 to illustrate some of the important characteristics of the B-spline curves. In this example, six control points are given. They are $\mathbf{P}_0 = [1,0]$, $\mathbf{P}_1 = [0,1]$, $\mathbf{P}_2 = [0,2]$, $\mathbf{P}_3 = [1,4]$, $\mathbf{P}_4 = [1,6]$, and $\mathbf{P}_5 = [-3,8]$. Therefore, for this curve,

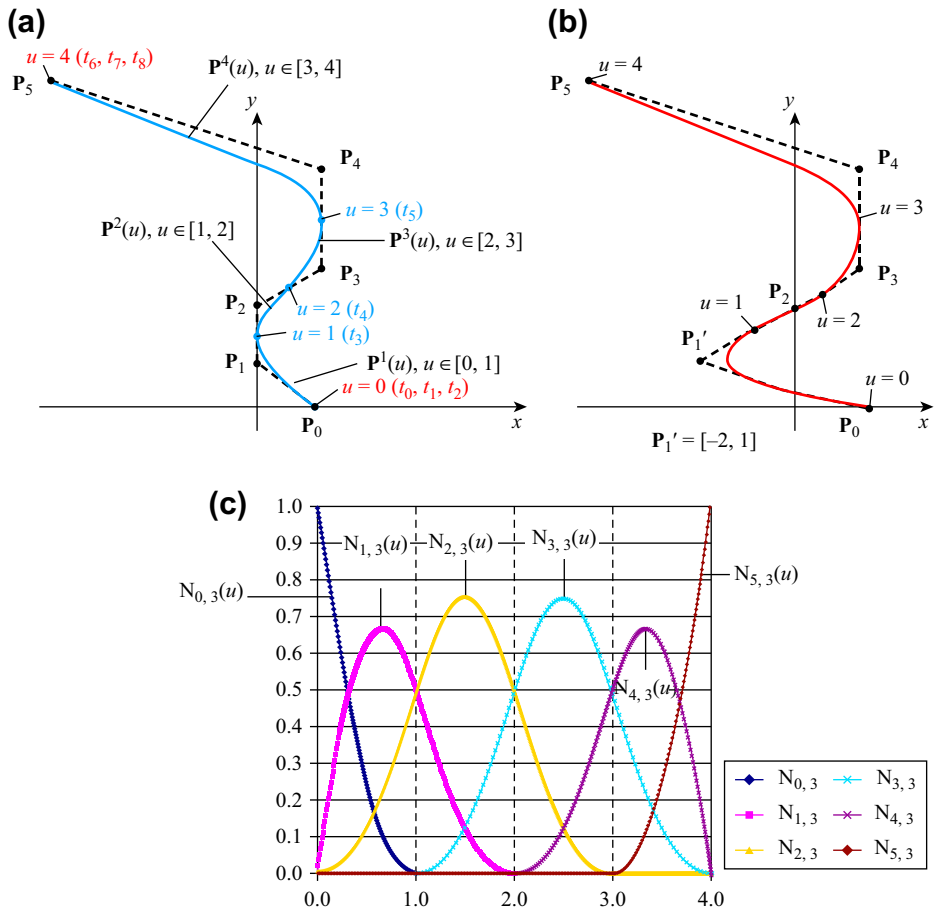


FIGURE 2.12

Quadratic B-spline curve. (a) A curve defined by six control points. (b) Geometric shape of the curve altered locally by relocating a control point $\mathbf{P}_1 = [0,1]$ to $\mathbf{P}_1{}' = [-2,1]$. (c) Quadratic basis functions.

$n = 5, k = 3, u \in [0, (5+1)-(3-1) = 4]$, and we have $n + k + 1 = 5 + 3 + 1 = 9$ knots. These knots are defined, according to Eq. 2.46, as

$$\begin{aligned} t_{0,1,2} &= 0 \\ t_3 &= 1 \\ t_4 &= 2 \\ t_5 &= 3 \\ t_{6,7,8} &= 4. \end{aligned} \tag{2.47}$$

Note that $t_{0,1,2} = 0$ and $t_{6,7,8} = 4$ are repeated knots, and $t_3 = 1$, $t_4 = 2$, and $t_5 = 3$ are nonrepeated.

The basis functions, according to Eqs 2.44 and 2.45, can be derived as follows (for details, see Appendix 2A):

$$\mathrm{N}_{0,3}(u) = (1-u)^2, \quad 0 \le u \le 1 \tag{2.48a}$$

$$\mathrm{N}_{1,3}(u) = \begin{cases} \frac{1}{2}u(4-3u), & 0 \le u \le 1 \\ \frac{1}{2}(2-u)^2, & 1 \le u \le 2 \end{cases} \tag{2.48b}$$

$$\mathrm{N}_{2,3}(u) = \begin{cases} \frac{1}{2}u^2, & 1 \le u \le 2 \\ \frac{1}{2}\left(-2u^2+6u-3\right), & 2 \le u \le 3 \\ \frac{1}{2}(3-u)^2, & 3 \le u \le 4 \end{cases} \tag{2.48c}$$

$$\mathrm{N}_{3,3}(u) = \begin{cases} \frac{1}{2}(u-1)^2, & 1 \le u \le 2 \\ \frac{1}{2}\left(-2u^2+10u-11\right), & 2 \le u \le 3 \\ \frac{1}{2}(4-u)^2, & 3 \le u \le 4 \end{cases} \tag{2.48d}$$

$$\mathrm{N}_{4,3}(u) = \begin{cases} \frac{1}{2}(u-2)^2, & 2 \le u \le 3 \\ \frac{1}{2}\left(-3u^2+20u-32\right), & 3 \le u \le 4 \end{cases} \tag{2.48e}$$

$$\mathrm{N}_{5,3}(u) = (u-3)^2, \quad 3 \le u \le 4. \tag{2.48f}$$

Hence from Eq. 2.43, the B-spline curve can be written as

$$\begin{aligned} \mathbf{P}(u) &= \sum_{i=0}^{5} \mathbf{P}_i \mathrm{N}_{i,k}(u) \\ &= \mathbf{P}_0\mathrm{N}_{0,3}(u) + \mathbf{P}_1\mathrm{N}_{1,3}(u) + \mathbf{P}_2\mathrm{N}_{2,3}(u) + \mathbf{P}_3\mathrm{N}_{3,3}(u) + \mathbf{P}_4\mathrm{N}_{4,3}(u) + \mathbf{P}_5\mathrm{N}_{5,3}(u) \\ &= \begin{cases} (1-u)^2\mathbf{P}_0 + \frac{1}{2}u(4-3u)\mathbf{P}_1 + \frac{1}{2}u^2\mathbf{P}_2, & 0 \le u \le 1 \\ \frac{1}{2}(2-u)^2\mathbf{P}_1 + \frac{1}{2}\left(-2u^2+6u-3\right)\mathbf{P}_2 + \frac{1}{2}(u-1)^2\mathbf{P}_3, & 1 \le u \le 2 \\ \frac{1}{2}(3-u)^2\mathbf{P}_2 + \frac{1}{2}\left(-2u^2+10u-11\right)\mathbf{P}_3 + \frac{1}{2}(u-2)^2\mathbf{P}_4, & 2 \le u \le 3 \\ \frac{1}{2}(4-u)^2\mathbf{P}_3 + \frac{1}{2}\left(-3u^2+20u-32\right)\mathbf{P}_4 + \frac{1}{2}(u-3)^2\mathbf{P}_5, & 3 \le u \le 4 \end{cases} \end{aligned} \tag{2.49}$$

There are several important observations:

1. As illustrated in Eq. 2.49, there are actually four piecewise quadratic B-spline curve segments joined with C^1-continuity at the nonrepeated knots, that is, at $t_3 = 1$, $t_4 = 2$, and $t_5 = 3$. You may want to verify the statement regarding the C^1-continuity by taking derivatives of the curve segment equations and plugging in respective u values at the junctions.
2. The B-spline curve starts and ends at the first and last control points, respectively, and is tangent to the first and the last line segments of the control polygon, respectively, similar to that of a Bézier curve.
3. The knot $t_3 = 1$ (where $u = 1$) is the midpoint of the line segment $\mathbf{P}_1\mathbf{P}_2$ of the control polygon, which is also the junction of the first and second B-spline curve segments, $\mathbf{P}^1(u)$ and $\mathbf{P}^2(u)$, respectively. The same is true for other nonrepeated knots. For a quadratic B-spline curve, its curve segments touch the midpoint of their respective line segments of the control polygon, and are tangent to the line segments at the contact points. Note that this is only true for quadratic curves. Increasing the polynomial order of a B-spline curve results in "pulling" the curve away from its control polygon, as illustrated in Figure 2.13 with a quadratic and a cubic B-spline curves.
4. As revealed in Eq. 2.49, the four curve segments are controlled by their respective control polygons. For example, curve segment 1, $\mathbf{P}^1(u)$ with $u \in [0,1]$, is controlled by polygon $\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2$; curve segment 2, $\mathbf{P}^2(u)$ with $u \in [1,2]$, is controlled by polygon $\mathbf{P}_1\mathbf{P}_2\mathbf{P}_3$; and so on. Therefore, when a control point is relocated, for example, moving control point $\mathbf{P}_1 = [0,1]$ to $\mathbf{P}_1' = [-2,1]$, as shown in Figure 2.12(b), instead of the entire curve, only curve segments 1 and 2 are affected. These local-control characteristics are desirable for fine-tuning the local geometric shape for part design.

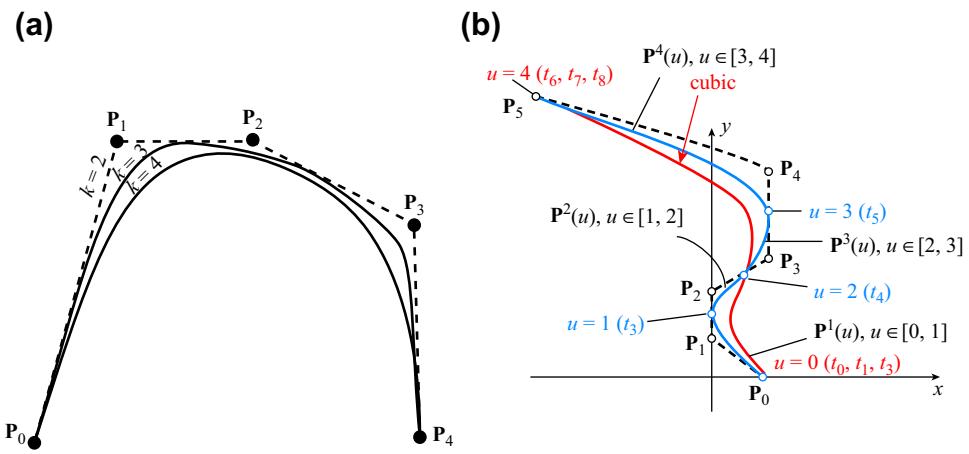


FIGURE 2.13

B-spline curves being "pushed" away from the control polygon when increasing their polynomial order. (a) $k = 2$ linear, $k = 3$ quadratic, and $k = 4$ cubic curves. (b) Quadratic and cubic B-spline curves defined by the same set of control points.

5. The B-spline curve and Bézier curve are identical when $(n + 1) - (k - 1) = n - k + 2 = 1$, implying there is only one curve segment in the B-spline curve. In this case, $n = k - 1$; that is, the n and k are coupled and the polynomial order of the curve $(k - 1)$ equals n, which is the number of control points minus 1. As discussed before, adding control points to a Bézier curve increases its polynomial order. However, adding control points to a B-spline curve increases the number of curve segments (and hence the parameter u domain) while keeping the same polynomial order. This is one of the most desirable characteristics of the B-spline curve in geometric modeling.
6. As illustrated in Figure 2.12(c), the six basis functions are symmetric in pairs. For example, $N_{0,3}(u)$ and $N_{5,3}(u)$, and $N_{1,3}(u)$ and $N_{4,3}(u)$ are symmetric. $N_{3,3}(u)$ and $N_{4,3}(u)$ are not only symmetric, but their geometric shape is in fact identical. The basis functions $N_{3,3}(u)$ and $N_{4,3}(u)$ are referred to as uniform, and are employed to construct uniform B-spline curves to be discussed next. The remaining four basis functions are nonuniform. Uniform or nonuniform basis functions stem from the spans of the knots. As shown in Eq. 2.44, the basis functions are defined recursively, with the lowest-order functions, step functions, defined in Eq. 2.45. All basis functions are strongly influenced by knots. As illustrated in Eq. 2.47, three knots repeat at 0 and another three repeat at 4. The spans between consecutive knots are nonuniform; that is, some are 0 (between the repeated knots) and some are 1. According to Eq. 2.45, basis functions $N_{0,1} = N_{1,1} = N_{6,1} = N_{7,1} = 0$ due to the zero-span between repeated knots. As shown in Figure 2.14, such zero-span affects basis functions all the way to $N_{0,3}(u)$, $N_{1,3}(u)$, $N_{4,3}(u)$, and $N_{5,3}(u)$, which are called nonuniform. B-spline curves constructed by using the basis functions, including nonuniform ones, are called nonuniform B-spline curves. One important characteristic of a nonuniform B-spline curve is that the curve starts and ends at the first and last control points, and it is tangent at the respective end control points to the control polygon.
7. Similar to the Bernstein polynomials, the sum of all basis functions is 1; that is, $\sum_{i=0}^{n} N_{i,k}(u) = 1$, implying that the basis functions form a partition of unity. In addition, $0 \leq N_{i,k}(u) \leq 1$; therefore, the convex hull property prevails, which ensures that the curve lies within the convex hull of the control points, just like a Bézier curve.

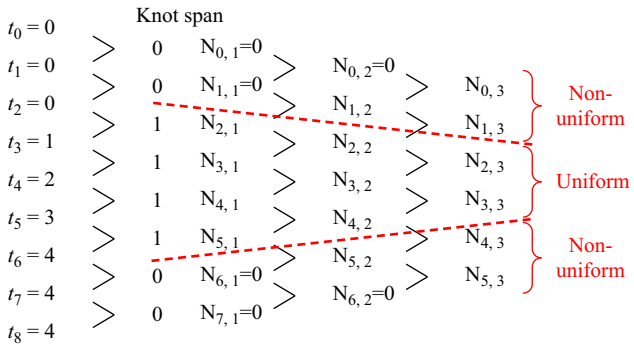


FIGURE 2.14

Uniform and nonuniform basis functions stem from the uniform and nonuniform spans between neighboring knots.

2.2.5.2 Uniform B-Spline Curves

A uniform B-spline curve is constructed by only using the uniform basis functions. As discussed in the earlier example, the basis functions $N_{2,3}(u)$ and $N_{3,3}(u)$, shown in Eqs 2.48c and 2.48d, respectively, are uniform. A B-spline curve constructed only by using such functions is called a uniform B-spline curve. By closely examining Eqs 2.48c and 2.48d, the uniform basis functions can be generalized for quadratic curves as the following:

$$N_{i,3}(u) = \begin{cases} \frac{1}{2}(-u+i)^2 \\ \frac{1}{2}[(u-i+2)(-u+i)+(-u+i+1)(u-i+1)], \quad u\in[i-1,i]. \\ \frac{1}{2}(u-i+1)^2 \end{cases} \tag{2.50}$$

A B-spline curve segment can then be constructed as

$$\begin{aligned} \mathbf{P}^i(u) = {} & \frac{1}{2}(-u+i)^2\mathbf{P}_{i-1} + \frac{1}{2}[(u-i+2)(-u+i)+(-u+i+1)(u-i+1)]\mathbf{P}_i \\ & + \frac{1}{2}(u-i+1)^2\mathbf{P}_{i+1}, \quad u\in[i-1,i]. \end{aligned} \tag{2.51}$$

By replacing u with $u+i-1$ in Eq. 2.51 to eliminate the i in the parentheses on the right hand side of Eq. 2.51, we have

$$\mathbf{P}^i(u) = \frac{1}{2}(1-u)^2\mathbf{P}_{i-1} + \frac{1}{2}\left(-2u^2+2u+1\right)\mathbf{P}_i + \frac{1}{2}u^2\mathbf{P}_{i+1}, \quad u\in[0,1], \quad i\in[1,n-1] \tag{2.52}$$

where $n+1$ is the total number of control points. Note that a set of $n+1$ control points defines $n-1$ quadratic uniform B-spline curve segments by using Eq. 2.52.

It is apparent that Eq. 2.52 is more desirable for constructing uniform B-spline curves because the index i is removed in the basis functions and the u domain is converted back to [0,1]. Another advantage of Eq. 2.52 is that it can be written in a matrix form, similar to those discussed before; that is,

$$\mathbf{P}^i(u) = [u^2 \quad u \quad 1]\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \end{bmatrix} = \mathbf{U}_{1\times3}\mathbf{M}^3_{3\times3}\begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \end{bmatrix}, \quad u\in[0,1], i\in[1,n-1] \tag{2.53}$$

where $\mathbf{M}^3$ is a constant 3×3 matrix. Note that a 4×4 $\mathbf{M}^4$ matrix can be derived by following the same steps (left as an exercise) for a cubic uniform B-spline curve:

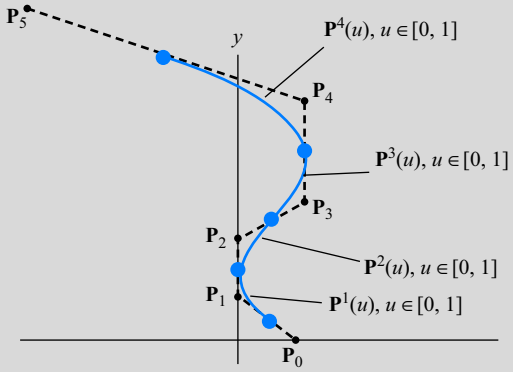
$$\mathbf{P}^i(u) = [u^3 \quad u^2 \quad u \quad 1]_{1\times4}\mathbf{M}^4_{4\times4}\begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \end{bmatrix} = \mathbf{U}_{1\times4}\mathbf{M}^4_{4\times4}\begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \end{bmatrix}, \quad u\in[0,1], i\in[1,n-2] \tag{2.54}$$

where

$$\mathbf{M}_{4\times4}^{4} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}. \tag{2.55}$$

EXAMPLE 2.8

Use the same six control points as before, $\mathbf{P}_0 = [1,0]$, $\mathbf{P}_1 = [0,1]$, $\mathbf{P}_2 = [0,2]$, $\mathbf{P}_3 = [1,4]$, $\mathbf{P}_4 = [1,6]$, and $\mathbf{P}_5 = [-3,8]$ to construct a quadratic uniform B-spline curve like the one shown below. Note that curve segment $\mathbf{P}^1(u)$ does not start from the control point $\mathbf{P}_0$ and curve segment $\mathbf{P}^4(u)$ does not end at the last control point $\mathbf{P}_5$. All neighboring curve segments join at their respective nonrepeated knots and are tangent to the control polygon at these knots.



Solutions

Using Eq. 2.53, the following equations describe the respective four curve segments:

$$\mathbf{P}^1(u) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix}\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix} = \begin{bmatrix} u^2 & u & 1 \end{bmatrix}\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}u^2 - u + \frac{1}{2}, & u + \frac{1}{2} \end{bmatrix}$$

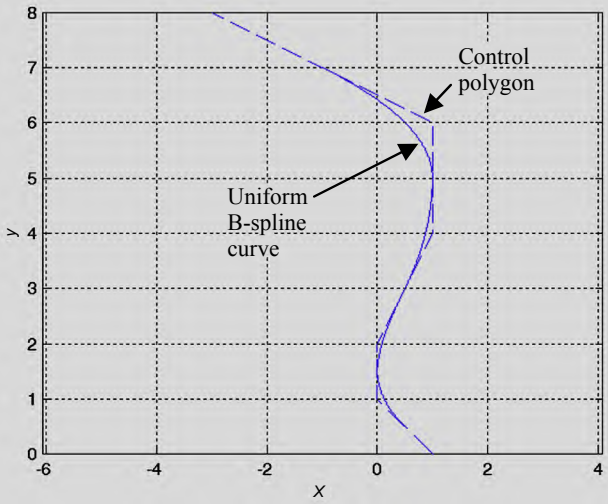
$$\mathbf{P}^2(u) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix}\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} = \begin{bmatrix} u^2 & u & 1 \end{bmatrix}\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 0 & 2 \\ 1 & 4 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}u^2, & \frac{1}{2}u^2 + u + \frac{3}{2} \end{bmatrix}$$

$$\mathbf{P}^3(u) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix}\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{bmatrix} = \begin{bmatrix} u^2 & u & 1 \end{bmatrix}\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 2 \\ 1 & 4 \\ 1 & 6 \end{bmatrix} = \begin{bmatrix} \frac{-1}{2}u^2 + u + \frac{1}{2}, & 2u + 3 \end{bmatrix}$$

$$\mathbf{P}^4(u) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix}\frac{1}{2}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{P}_3 \\ \mathbf{P}_4 \\ \mathbf{P}_5 \end{bmatrix} = \begin{bmatrix} u^2 & u & 1 \end{bmatrix}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 4 \\ 1 & 6 \\ -3 & 8 \end{bmatrix} = \begin{bmatrix} -2u^2 + 1, & 2u + 5 \end{bmatrix}$$

EXAMPLE 2.8—cont'd

The curves are graphed in Matlab with the script shown below.



Matlab Script:

```
ezplot('0.5*u^2-u+0.5','u+0.5',[0,1])
grid on
hold on
ezplot('0.5*u^2','0.5*u^2+u+1.5',[0,1])
ezplot('-0.5*u^2+u+0.5','2*u+3',[0,1])
ezplot('-2*u^2+1','2*u+5',[0,1])
Px=[1 0 0 1 1 -3]
Py=[0 1 2 4 6 8]
plot(Px,Py,'--')
```

2.2.5.3 Closed Uniform B-Spline Curves

The curve shown above is an open B-spline curve, in which the start and end control points do not coincide. Uniform B-spline curves are well suited for modeling part geometry of a smooth closed profile. In this case, its control polygon must be closed, which can be achieved by simply aligning the first and the last control points. For example, the six control points shown in Figure 2.15 form a closed control polygon by connecting control points $\mathbf{P}_5$ back to $\mathbf{P}_0$. A quadratic uniform B-spline curve can be constructed by using Eq. 2.53 as

$$\mathbf{P}^i(u) = \mathbf{U}_{1\times 3}\mathbf{M}^3_{3\times 3}\begin{bmatrix}\mathbf{P}_{(i-1)\bmod(n+1)}\\ \mathbf{P}_{(i)\bmod(n+1)}\\ \mathbf{P}_{(i+1)\bmod(n+1)}\end{bmatrix}, \quad u\in[0,1], i\in[1,n+1] \tag{2.56}$$

in which "mod" is the remaining operator. For example, if $i = 6$ and $n = 5$, then $(i-1)\bmod(n+1) = 5 \bmod 6 = 5$; $(i) \bmod(n+1) = 6 \bmod 6 = 0$; and $(i+1)\bmod(n+1) = 7 \bmod 6 = 1$. Therefore, from Eq. 2.56, the sixth curve segment shown in Figure 2.15 can be found as

$$\mathbf{P}^6(u) = \mathbf{U}_{1\times 3}\mathbf{M}^3_{3\times 3}\begin{bmatrix}\mathbf{P}_{(6-1)\bmod(5+1)}\\ \mathbf{P}_{(6)\bmod(5+1)}\\ \mathbf{P}_{(6+1)\bmod(5+1)}\end{bmatrix} = \mathbf{U}_{1\times 3}\mathbf{M}^3_{3\times 3}\begin{bmatrix}\mathbf{P}_5\\ \mathbf{P}_0\\ \mathbf{P}_1\end{bmatrix}.$$

The mod operator is simply introduced to manage the index of the control points as well as adding curve segments to form a closed loop.

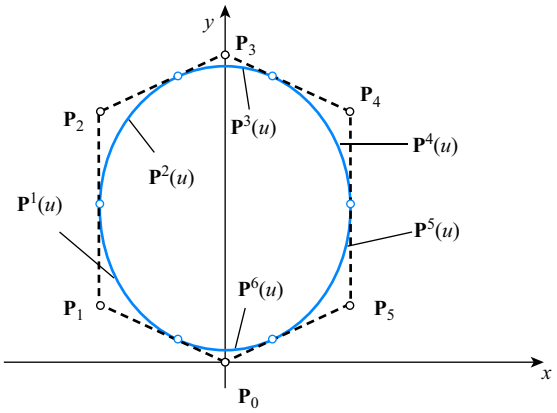


FIGURE 2.15

A closed control polygon of six control points that encloses six closed uniform B-spline curve segments.

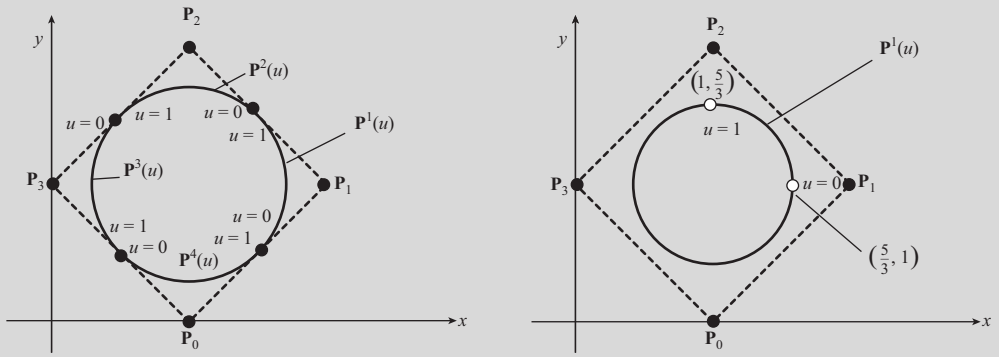
Similarly, for cubic curves, we have

$$\mathbf{P}^i(u) = \mathbf{U}_{1\times 4}\mathbf{M}^4_{4\times 4}\begin{bmatrix}\mathbf{P}_{(i-1)\bmod(n+1)}\\ \mathbf{P}_{(i)\bmod(n+1)}\\ \mathbf{P}_{(i+1)\bmod(n+1)}\\ \mathbf{P}_{(i+2)\bmod(n+1)}\end{bmatrix}, \quad u \in [0,1], i \in [1, n+1]. \tag{2.57}$$

The following example illustrates the characteristics of the closed uniform B-spline curves in more detail, using both quadratic and cubic curves.

EXAMPLE 2.9

Use the four control points, $\mathbf{P}_0 = [1,0]$, $\mathbf{P}_1 = [2,1]$, $\mathbf{P}_2 = [1,2]$, and $\mathbf{P}_3 = [0,1]$, which form a closed control polygon, to construct both a quadratic and a cubic uniform B-spline curve, similar to those shown below (left: closed quadratic uniform B-spline curve; right: closed cubic uniform B-spline curve).



EXAMPLE 2.9—cont'd

Using Eq. 2.56 for a quadratic curve, we have

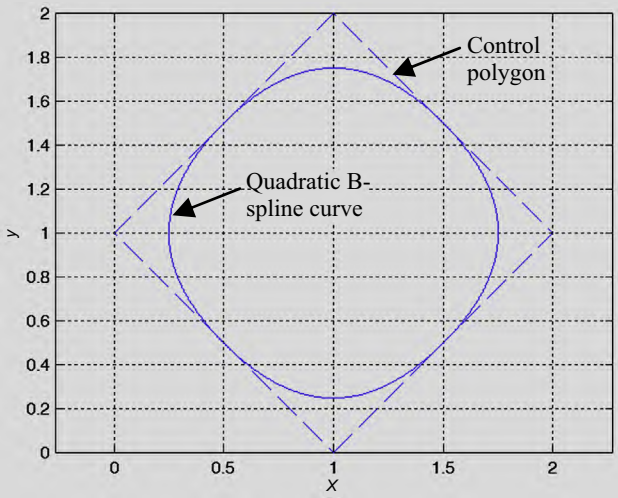
$$\mathbf{P}^1(u) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix} = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 1 & 2 \end{bmatrix} = \left[-u^2 + u + \frac{3}{2}, u + \frac{1}{2} \right]$$

$$\mathbf{P}^2(u) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 0 & 1 \end{bmatrix} = \left[-u + \frac{3}{2}, -u^2 + u + \frac{3}{2} \right]$$

$$\mathbf{P}^3(u) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_0 \end{bmatrix} = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \left[u^2 - u + \frac{1}{2}, -u + \frac{3}{2} \right]$$

$$\mathbf{P}^4(u) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_3 \\ \mathbf{P}_0 \\ \mathbf{P}_1 \end{bmatrix} = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} = \left[u + \frac{1}{2}, u^2 - u + \frac{1}{2} \right]$$

The curve is graphed in Matlab with the script shown below.



Matlab Script:

```
ezplot('-u^2+u+1.5','u+0.5',[0,1])
grid on
hold on
ezplot('-u+1.5','-u^2+u+1.5',[0,1])
ezplot('u^2-u+0.5','-u+1.5',[0,1])
ezplot('u+0.5','u^2-u+0.5',[0,1])
Px=[1 2 1 0 1]
Py=[0 1 2 1 0]
plot(Px,Py,'--')
```

Now, we use Eq. 2.57 for a cubic curve.

Continued

EXAMPLE 2.9—cont'd

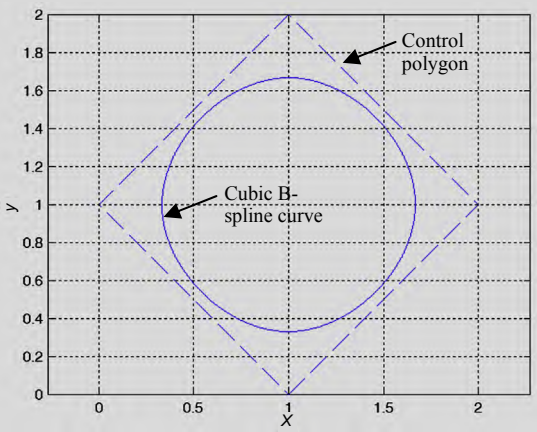
$$\mathbf{P}^1(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathrm{M}^4 \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_4 \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 1 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{3}u^3 - u^2 + \frac{5}{3}, & -\frac{1}{3}u^3 + u + 1 \end{bmatrix}$$

$$\mathbf{P}^2(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathrm{M}^4 \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_0 \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{3}u^3 - u + 1, & \frac{1}{3}u^3 - u^2 + \frac{5}{3} \end{bmatrix}$$

$$\mathbf{P}^3(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathrm{M}^4 \begin{bmatrix} \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_0 \\ \mathbf{P}_1 \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} \frac{-1}{3}u^3 + u^2 + \frac{1}{3}, & \frac{1}{3}u^3 - u + 1 \end{bmatrix}$$

$$\mathbf{P}^4(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \mathrm{M}^4 \begin{bmatrix} \mathbf{P}_3 \\ \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 2 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} \frac{-1}{3}u^3 + u + 1, & -\frac{1}{3}u^3 + u^2 + \frac{1}{3} \end{bmatrix}$$

The curve is graphed in Matlab with the script shown below.



Matlab Script:

```
ezplot('1/3*u^3-u^2+5/3','-1/3*u^3+u+1',[0,1])
grid on
hold on
ezplot('1/3*u^3-u+1','1/3*u^3-u^2+5/3',[0,1])
ezplot('-1/3*u^3+u^2+1/3','1/3*u^3-u+1',[0,1])
ezplot('-1/3*u^3+u+1','-
1/3*u^3+u^2+1/3',[0,1])
Px=[1 2 1 0 1]
Py=[0 1 2 1 0]
plot(Px,Py,'--')
```

As shown in this example, unlike the quadratic B-spline curve, the cubic curve does not contact the control polygon, as also indicated in Figure 2.13. The cubic B-spline is composed of four curve segments. How smooth is the cubic curve? Is the continuity C^1 or C^2 at the junctions of the curve segments? What are the Cartesian coordinates of the junction points of curve segments? For example, for curve segment 1, what are the locations of its start and end points? This is left as an exercise. The cubic curve looks much like a circle: Is it a true circle?

2.2.6 NURB CURVES

Can a B-spline curve represent a true circle? The answer is no. The polynomial order of a B-spline curve is finite. For a cubic curve, its polynomial order is 3; for a true circle, its polynomial order is infinite. Why infinite? As discussed in Section 2.2, a true circle on a plane can be represented in a polar coordinate system as

$$x = a + r\cos\theta, \quad \text{and} \quad y = b + r\sin\theta, \quad \theta \in [0, 2\pi]. \tag{2.58}$$

A Taylor series expansion for the trigonometry functions above, such as sin θ, is

$$\sin\theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \ldots = \frac{\sum_{n=1}^{\infty}(-1)^{n-1}\theta^{2n-1}}{(2n-1)!} \tag{2.59}$$

which is a function of an infinite polynomial order.

A parametric curve that is capable of representing geometric entities, such as a circle or any other conic curves, is NURB, which is one of the most versatile and general curves employed for geometric modeling.

Mathematically, a NURB curve is defined as

$$\mathbf{P}(u) = \frac{\sum_{i=0}^{n} \mathrm{h}_i \mathbf{P}_i \mathrm{N}_{i,k}(u)}{\sum_{i=0}^{n} \mathrm{h}_i \mathrm{N}_{i,k}(u)}, \quad u \in [0, (n+1) - (k-1)] \tag{2.60}$$

where $\mathrm{N}_{i,k}(u)$'s are the basis functions of the B-spline curve (discussed previously), $\mathbf{P}_i$ is the ith control point, h_i is the weight associated with the control point $\mathbf{P}_i$, and $n+1$ is the total number of control points.

Note that when all the weights are in unity (i.e., $\mathrm{h}_i = 1$), the NURB curve is no longer rational; in fact, it degenerates to a B-spline curve because the sum of the B-spline basis functions in the denominator of Eq. 2.60 is 1; i.e., $\sum_{i=0}^{n} \mathrm{h}_i \mathrm{N}_{i,k}(u) = \sum_{i=0}^{n} \mathrm{N}_{i,k}(u) = 1$.

In addition, the weights h_i play a significant role in determining the geometric shape of the NURB curve. For example, for a quadratic NURB of $n = 2$ and $k = 3$ shown in Figure 2.16(a), we set $\mathrm{h}_0 = \mathrm{h}_2 = 1$, and vary h_1. When $\mathrm{h}_1 = 0$, the curve becomes a straight line connecting $\mathbf{P}_0$ and $\mathbf{P}_2$. When h_1 is increased with a positive value, the curve is "pulled" closer to the control polygon. A negative h_1 value is "pushing" the curve to the opposite of the control polygon. Note that the convex hull property does not hold if $\mathrm{h}_1 < 0$.

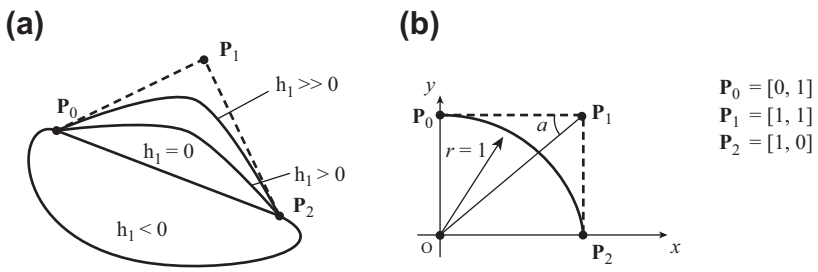


FIGURE 2.16

Quadratic NURB curve. (a) Effect of the weight h_1 to the geometric shape of the curve. (b) Representing a 90-degree circular arc.

Now, is it possible to find a value for the weight h_1 that allows the quadratic NURB curve enclosed by the control polygon to analytically represent a 90° circular arc of radius 1? The answer is yes (e.g., see Figure 2.16(b)). Let's take a look at the NURB curve in Example 2.10.

EXAMPLE 2.10

Use the three control points, $\mathbf{P}_0 = [0,1]$, $\mathbf{P}_1 = [1,1]$, and $\mathbf{P}_2 = [1,0]$ shown in Figure 2.16(b), which form a control polygon to construct a quadratic NURB curve that represents a 90-degree circular arc of radius 1 analytically.

Solutions

Any conic (including circles) can be parameterized in terms of rational quadratic functions. Hence, an arc of a conic has a NURB representation (Piegl and Tiller, 1987). It is shown mathematically in Appendix 2B that when using a quadratic NURB curve enclosed by a control polygon $\mathbf{P}_0\mathbf{P}_1\mathbf{P}_2$ to represent a circular arc, the weight h_1 is determined as $h_1 = \sin a$, where a is the angle of $\mathbf{P}_0\mathbf{P}_1\mathbf{O}$ shown in Figure 2.16(b). Therefore, for this example, the angle a is 45°; therefore, $h_1 = \sin a = \frac{1}{\sqrt{2}}$, and the NURB curve can be written using Eq. 2.60 as

$$\mathbf{P}(u) = \frac{\sum_{i=0}^{2} h_i \mathbf{P}_i N_{i,3}(u)}{\sum_{i=0}^{2} h_i N_{i,3}(u)} = \frac{h_0\mathbf{P}_0 N_{0,3}(u) + h_1\mathbf{P}_1 N_{1,3}(u) + h_2\mathbf{P}_2 N_{2,3}(u)}{h_0 N_{0,3}(u) + h_1 N_{1,3}(u) + h_2 N_{2,3}(u)}$$

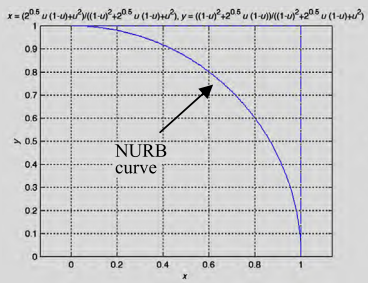
$$= \frac{(1)[0\ \ 1](1-u)^2 + \frac{1}{\sqrt{2}}[1\ \ 1]2u(1-u) + (1)[1\ \ 0]u^2}{(1)(1-u)^2 + \frac{1}{\sqrt{2}}2u(1-u) + (1)u^2}.$$

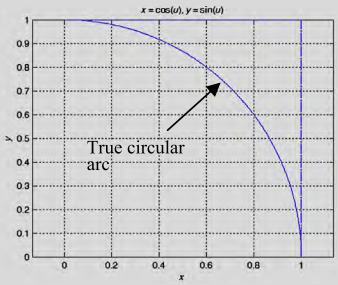
And in component form, we have

$$P_x(u) = \frac{\sqrt{2}u(1-u) + u^2}{(1-u)^2 + \sqrt{2}u(1-u) + u^2}$$

$$P_y(u) = \frac{(1-u)^2 + \sqrt{2}u(1-u)}{(1-u)^2 + \sqrt{2}u(1-u) + u^2}.$$

The NURB curve and a true circular arc are graphed in Matlab with the script shown below.






```
Matlab Script:

ezplot('(2^0.5*u*(1-u)+u^2)/((1-
u)^2+2^0.5*u*(1-u)+u^2)', '((1-
u)^2+2^0.5*u*(1-u))/((1-
u)^2+2^0.5*u*(1-u)+u^2)',[0,1])
grid on
hold on
Px=[0 1 1]
Py=[1 1 0]
plot(Px,Py,'--')
ezplot('cos(u)', 'sin(u)',[0,3.14159/2])
```

2.3 PARAMETRIC SURFACES

A parametric surface is a surface in the Euclidean space R^3, which is defined by parametric equations with two parameters (u,w). Parametric representation is probably the most general way to specify a surface. The curvature and arc length of curves on the surface, surface area, differential geometric

invariants such as the first and second fundamental forms, Gaussian, mean, and principal curvatures can all be computed from a given parameterization. Due to their generality, parametric surfaces are widely adopted in geometric modeling for support of product design and manufacturing, among many other applications.

In this section, we discuss several parametric surfaces that are commonly found in geometric modeling. In the next section (Section 2.4), we focus on CAD-generated surfaces represented in parametric form.

2.3.1 PARAMETRIC REPRESENTATION

Similar to parametric curves, a parametric surface in space can be written in the following parametric form:

$$\mathbf{S}(u,w) = \left[\mathrm{S}_x(u,w), \mathrm{S}_y(u,w), \mathrm{S}_z(u,w)\right]_{1\times 3}, \quad (u,w) \in [0,1] \times [0,1] \tag{2.61}$$

where u and w are the parameters or parametric coordinates of the surface. Usually, these parametric coordinates range between 0 and 1.

2.3.1.1 Bicubic Surface Patch

A bicubic surface patch can be defined in terms of cubic polynomials as

$$\begin{aligned}\mathbf{S}(u,w) &= \sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{a}_{ij}u^i w^j \\ &= \mathbf{a}_{33}u^3w^3 + \mathbf{a}_{32}u^3w^2 + \mathbf{a}_{31}u^3w + \mathbf{a}_{30}u^3 + \mathbf{a}_{23}u^2w^3 + \mathbf{a}_{22}u^2w^2 + \mathbf{a}_{21}u^2w \\ &\quad + \mathbf{a}_{20}u^2 + \mathbf{a}_{13}uw^3 + \mathbf{a}_{12}uw^2 + \mathbf{a}_{11}uw + \mathbf{a}_{10}u + \mathbf{a}_{03}w^3 + \mathbf{a}_{02}w^2 + \mathbf{a}_{01}w \\ &\quad + \mathbf{a}_{00}, (u,w) \in [0,1] \times [0,1]\end{aligned} \tag{2.62}$$

where $\mathbf{a}_{ij}$ is a 1×3 vector; that is, $\mathbf{a}_{ij} = [a_{ijx}, a_{ijy}, a_{ijz}]$. Hence, for example, the x-component of a parametric surface is

$$\begin{aligned}\mathrm{S}_x(u,w) &= \sum_{i=0}^{3}\sum_{j=0}^{3}a_{ijx}u^i w^j \\ &= a_{33x}u^3w^3 + a_{32x}u^3w^2 + a_{31x}u^3w + a_{30x}u^3 \\ &\quad + a_{23x}u^2w^3 + a_{22x}u^2w^2 + a_{21x}u^2w + a_{20x}u^2 \\ &\quad + a_{13x}uw^3 + a_{12x}uw^2 + a_{11x}uw + a_{10x}u \\ &\quad + a_{03x}w^3 + a_{02x}w^2 + a_{01x}w + a_{00x} \\ &= \begin{bmatrix}u^3 & u^2 & u & 1\end{bmatrix}\begin{bmatrix}a_{33x} & a_{32x} & a_{31x} & a_{30x} \\ a_{23x} & a_{22x} & a_{21x} & a_{20x} \\ a_{13x} & a_{12x} & a_{11x} & a_{10x} \\ a_{03x} & a_{02x} & a_{01x} & a_{00x}\end{bmatrix}\begin{bmatrix}w^3 \\ w^2 \\ w \\ 1\end{bmatrix} \\ &= \mathbf{U}_{1\times 4}\mathbf{A}x_{4\times 4}\mathbf{W}^{\mathrm{T}}_{4\times 1}, \quad (u,w) \in [0,1] \times [0,1]\end{aligned} \tag{2.63a}$$

where $\mathbf{A}_x$ is a 4×4 matrix of 16 coefficients, which are to be determined. Similarly,

$$\begin{aligned} S_y(u,w) &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_{33y} & a_{32y} & a_{31y} & a_{30y} \\ a_{23y} & a_{22y} & a_{21y} & a_{20y} \\ a_{13y} & a_{12y} & a_{11y} & a_{10y} \\ a_{03y} & a_{02y} & a_{01y} & a_{00y} \end{bmatrix} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix} \\ &= \mathbf{U}_{1\times4}\mathbf{A}y_{4\times4}\mathbf{W}^{\mathrm{T}}_{4\times1}, \quad (u,w)\in[0,1]\times[0,1] \end{aligned} \tag{2.63b}$$

and

$$\begin{aligned} S_z(u,w) &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_{33z} & a_{32z} & a_{31z} & a_{30z} \\ a_{23z} & a_{22z} & a_{21z} & a_{20z} \\ a_{13z} & a_{12z} & a_{11z} & a_{10z} \\ a_{03z} & a_{02z} & a_{01z} & a_{00z} \end{bmatrix} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix} \\ &= \mathbf{U}_{1\times4}\mathbf{A}z_{4\times4}\mathbf{W}^{\mathrm{T}}_{4\times1}, \quad (u,w)\in[0,1]\times[0,1]. \end{aligned} \tag{2.63c}$$

Similarly, Eq. 2.62 can be written in a matrix form as

$$\begin{aligned} \mathbf{S}(u,w) &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_{33} & \mathbf{a}_{32} & \mathbf{a}_{31} & \mathbf{a}_{30} \\ \mathbf{a}_{23} & \mathbf{a}_{22} & \mathbf{a}_{21} & \mathbf{a}_{20} \\ \mathbf{a}_{13} & \mathbf{a}_{12} & \mathbf{a}_{11} & \mathbf{a}_{10} \\ \mathbf{a}_{03} & \mathbf{a}_{02} & \mathbf{a}_{01} & \mathbf{a}_{00} \end{bmatrix} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix} \\ &= \mathbf{U}_{1\times4}\mathbf{A}_{4\times4\times3}\mathbf{W}^{\mathrm{T}}_{4\times1}, \quad (u,w)\in[0,1]\times[0,1] \end{aligned} \tag{2.64}$$

where $\mathbf{A}$ is a 4×4×3 matrix of 48 coefficients (or a tensor of order 2), which are to be determined. Note that in Eq. 2.64, the sizes of vectors and matrix do not match; thus, the multiplications cannot be actually carried out. We simply use the equation to describe the parametric surface in a more compact form. When performing multiplications, the x-, y-, and z-components of the surface equations in Eq. 2.61 must be carried out separately (e.g., like that of Eq. 2.63a for the x-component of the surface).

2.3.1.2 16-Point Format

A bicubic surface patch can be created by 16 distinct points arranged in a 4×4 matrix form, as shown in Figure 2.17(a). Similar to the cubic spline curve, these points are assumed at the 0, 1/3, 2/3, and 1 locations of the parametric coordinates u and w; hence, the surface equation can be written as

$$\mathbf{S}(u,w) = \mathbf{U}\mathbf{N}^{\mathrm{s}}\mathbf{G}^{\mathrm{s}}\mathbf{N}^{\mathrm{s}^{\mathrm{T}}}\mathbf{W}^{\mathrm{T}}, \quad (u,w)\in[0,1]\times[0,1] \tag{2.65}$$

where

$$\mathbf{N}^{\mathrm{s}} = \begin{bmatrix} -9/2 & 27/2 & -27/2 & 9/2 \\ 9 & -45/2 & 18 & -9/2 \\ -11/2 & 9 & -9/2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.66}$$

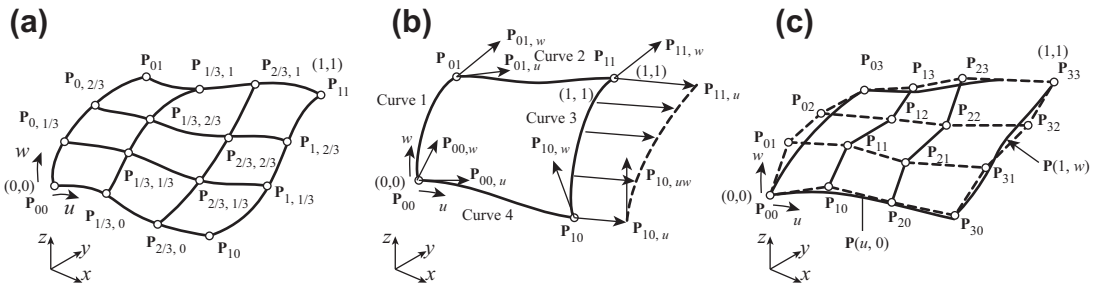


FIGURE 2.17

Bicubic surface patches as defined by (a) 16 distinct points; (b) corner points, tangents vectors, and twist vectors (Coons patch); and (c) 16 control points (Bézier surface patch).

which is identical to that of the cubic spline curve, and

$$\mathbf{G}^{\mathrm{S}} = \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{bmatrix}_{4\times4\times3} = \begin{bmatrix} \mathbf{P}(0,0) & \mathbf{P}\left(0,\frac{1}{3}\right) & \mathbf{P}\left(0,\frac{2}{3}\right) & \mathbf{P}(0,1) \\ \mathbf{P}\left(\frac{1}{3},0\right) & \mathbf{P}\left(\frac{1}{3},\frac{1}{3}\right) & \mathbf{P}\left(\frac{1}{3},\frac{2}{3}\right) & \mathbf{P}\left(\frac{1}{3},1\right) \\ \mathbf{P}\left(\frac{2}{3},0\right) & \mathbf{P}\left(\frac{2}{3},\frac{1}{3}\right) & \mathbf{P}\left(\frac{2}{3},\frac{2}{3}\right) & \mathbf{P}\left(\frac{2}{3},1\right) \\ \mathbf{P}(1,0) & \mathbf{P}\left(1,\frac{1}{3}\right) & \mathbf{P}\left(1,\frac{2}{3}\right) & \mathbf{P}(1,1) \end{bmatrix}_{4\times4\times3} \tag{2.67}$$

which is defined by the Cartesian coordinates of the 16 points.

2.3.1.3 Coons Patch

A Coons patch (named after Steven Anson Coons, 1912–1979) is a bicubic parametric surface formed by four corner points, eight tangent vectors (two vectors in the u and w directions, respectively, at each of the four corners), and four twister vectors at the respective four corner points, as shown in Figure 2.17(b).

Mathematically, a Coons patch is defined as

$$\mathbf{S}(u,w) = \mathbf{U}\mathbf{N}^{\mathrm{v}}\mathbf{G}^{\mathrm{v}}\mathbf{N}^{\mathrm{v}^{\mathrm{T}}}\mathbf{W}^{\mathrm{T}}, \quad (u,w) \in [0,1] \times [0,1] \tag{2.68}$$

where

$$\mathbf{N}^{\mathrm{v}} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.69}$$

which is identical to that of the Hermit cubic curve (two point and two vector format), and

$$\mathbf{G}^{v} = \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{00,w} & \mathbf{P}_{01,w} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{10,w} & \mathbf{P}_{11,w} \\ \mathbf{P}_{00,u} & \mathbf{P}_{01,u} & \mathbf{P}_{00,uw} & \mathbf{P}_{01,uw} \\ \mathbf{P}_{10,u} & \mathbf{P}_{11,u} & \mathbf{P}_{10,uw} & \mathbf{P}_{11,uw} \end{bmatrix}_{4\times4\times3} \tag{2.70}$$

where $\mathbf{P}_{00} = \mathbf{S}(0,0)$, $\mathbf{P}_{01} = \mathbf{S}(0,1)$, $\mathbf{P}_{10} = \mathbf{S}(1,0)$, and $\mathbf{P}_{11} = \mathbf{S}(1,1)$ are the four corner points; $\mathbf{P}_{00,u} = \partial\mathbf{S}/\partial u|_{u=w=0}$, $\mathbf{P}_{01,u}$, $\mathbf{P}_{10,u}$, and $\mathbf{P}_{11,u}$ are the tangent vectors in the u direction at the four corner points; $\mathbf{P}_{00,w} = \partial\mathbf{S}/\partial w|_{u=w=0}$, $\mathbf{P}_{01,w}$, $\mathbf{P}_{10,w}$, and $\mathbf{P}_{11,w}$ are the tangent vectors in the w direction at the four corner points; and $\mathbf{P}_{00,uw} = \partial^2\mathbf{S}/\partial u\partial w|_{u=w=0}$, $\mathbf{P}_{01,uw}$, $\mathbf{P}_{10,uw}$, and $\mathbf{P}_{11,uw}$ are the twister vectors at the four corner points.

Note that a twister vector represents changes of tangent vector in u (or w) direction at a corner point along a boundary curve in the w (or u) direction. For example, $\mathbf{P}_{10,uw} = \partial/\partial w(\partial\mathbf{S}/\partial u)|_{u=1,\ w=0} = \partial\mathbf{P}_{10,u}/\partial w$ is the derivative of the tangent vector along the u direction at $\mathbf{P}_{10}$ with respect to w (i.e., along boundary curve 3 shown in Figure 2.17(b)). Geometrically, this twister vector represents the changes of the tangent vector $\mathbf{P}_{10,u}$ along boundary curve 3, as shown in Figure 2.17(b). The same twister vector can also be interpreted as $\mathbf{P}_{10,uw} = \partial/\partial u(\partial\mathbf{S}/\partial w)|_{u=1,\ w=0} = \partial\mathbf{P}_{10,w}/\partial u$, which is the derivative of the tangent vector along the w direction at $\mathbf{P}_{10}$ with respect to u, representing the changes of the tangent vector $\mathbf{P}_{10,w}$ along boundary curve 4. Also, the first two rows of the matrix $\mathbf{G}^{v}$ are boundary curves 1 and 3, respectively; and columns 1 and 2 are boundary curves 4 and 2, respectively.

C^0-continuity of composite Coons patches can be imposed by joining their neighboring boundary edges. For example, to ensure C^0-continuity across the two patches A and B, the patches depicted in Figure 2.18(a) must have

$$\mathbf{S}^{A}(1,w) = \mathbf{S}^{B}(0,w), \quad \text{or} \quad \mathbf{P}^{A}_{10} = \mathbf{P}^{B}_{00}, \mathbf{P}^{A}_{11} = \mathbf{P}^{B}_{01}, \mathbf{P}^{A}_{10,w} = \mathbf{P}^{B}_{00,w}, \quad \text{and} \quad \mathbf{P}^{A}_{11,w} = \mathbf{P}^{B}_{01,w}. \tag{2.71}$$

For G^1-continuity, the tangent vectors across the joining boundary of the surfaces must be collinear; that is,

$$\mathbf{S}^{A}_{,u}(1,w) = C\mathbf{S}^{B}_{,u}(0,w), \quad \text{or} \quad \mathbf{P}^{A}_{10,u} = C\mathbf{P}^{B}_{00,u}, \mathbf{P}^{A}_{11,u} = C\mathbf{P}^{B}_{01,u}, \mathbf{P}^{A}_{10,uw} = C\mathbf{P}^{B}_{00,uw}, \quad \text{and} \quad \mathbf{P}^{A}_{11,uw} = C\mathbf{P}^{B}_{01,uw}, C \neq 0. \tag{2.72}$$

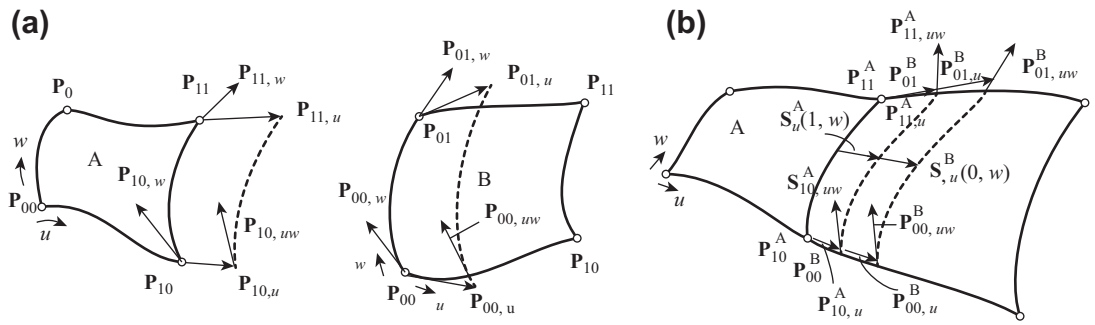


FIGURE 2.18

Continuity of composite Coons patches. (a) C^0-continuity. (b) G^1- or C^1-continuity.

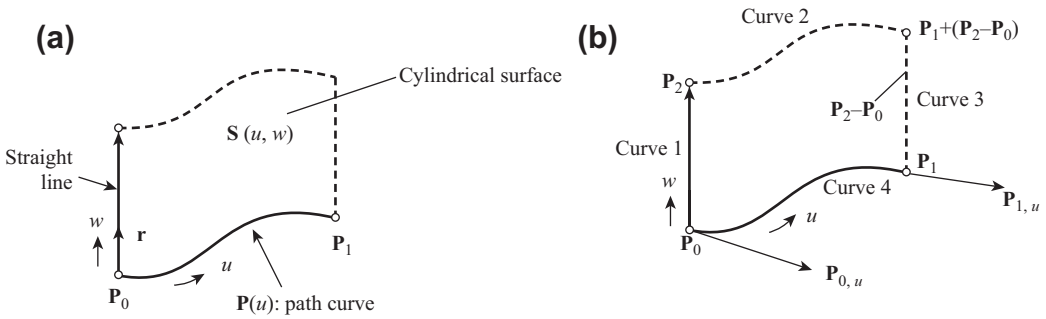


FIGURE 2.19

Representing a cylindrical surface using a Coons patch. (a) Cylindrical surface. (b) Coons patch representing the cylindrical surface.

For C^1-continuity, the constant $C = 1$.

The Coons patch is popular for support of geometric modeling. One key characteristic of the patch is that its geometric shape can be controlled or adjusted by not only altering the corner points, but also the tangent vectors and twister vectors. By imposing required properties on these vectors, we can designate a Coons patch to represent specific surfaces. For example, a cylindrical surface shown in Figure 2.19 can be represented by a Coons patch, which is illustrated next.

In CAD, a cylindrical surface is created when we extrude a sketch profile in the depth (or extrusion) direction. In geometric modeling, a cylindrical surface can be thought of as sweeping a straight line along a path curve, as shown in Figure 2.19(a), in which the path curve $\mathbf{P}(u)$ is assumed as a cubic curve, and the straight line is along the w direction, defined by a vector $\mathbf{r}$. A Coons patch that represents this cylindrical surface is shown in Figure 2.19(b), in which boundary curve 4 is the path curve and boundary curve 1 is the straight line.

The matrix $\mathbf{G}^{\mathrm{v}}$ that defines the cylindrical surface is written in Eq. 2.73, in which the first two rows define the straight boundary edges 1 and 3, respectively; and the first two columns are boundary curves 4 and 2, respectively. In fact, the first column of matrix $\mathbf{G}^{\mathrm{v}}$ is the path curve $\mathbf{P}(\mathrm{u})$, and the first row is the straight line that sweeps along the path curve. Note that all twister vectors are 0 because tangent vectors are not varying along any of the boundary edges.

$$\mathbf{G}^{\mathrm{v}} = \begin{bmatrix} \mathbf{P}_0 & \mathbf{P}_2 & \mathbf{P}_2 - \mathbf{P}_0 & \mathbf{P}_2 - \mathbf{P}_0 \\ \mathbf{P}_1 & \mathbf{P}_1 + \mathbf{P}_2 - \mathbf{P}_0 & \mathbf{P}_2 - \mathbf{P}_0 & \mathbf{P}_2 - \mathbf{P}_0 \\ \mathbf{P}_{0,u} & \mathbf{P}_{0,u} & 0 & 0 \\ \mathbf{P}_{1,u} & \mathbf{P}_{1,u} & 0 & 0 \end{bmatrix}_{4\times4\times3} \tag{2.73}$$

2.3.1.4 Bézier Surface

Mathematically, a Bézier surface (or patch) is defined as

$$\mathbf{S}(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{P}_{ij}\mathrm{B}_{i,n}(u)\mathrm{B}_{j,m}(w), \quad (u,w) \in [0,1] \times [0,1] \tag{2.74}$$

where $B_{i,n}(u)$ and $B_{j,m}(w)$ are Bernstein polynomials of order n and m, respectively, in u and w; and $\mathbf{P}_{ij}$ is the control point of the ith row at the jth location in the $(n+1)\times(m+1)$ control point matrix. Note that n does not have to be equal to m, implying that the polynomial orders of the Bézier surface along the u and w direction do not have to be identical.

As a special case, a bicubic Bézier surface is defined by 16 control points arranged in a 4×4 matrix form that forms a control polyhedron, as shown in Figure 2.17(c). From Eq. 2.74, a bicubic Bézier surface is then defined as

$$\mathbf{S}(u,w) = \sum_{i=0}^{3}\sum_{j=0}^{3} \mathbf{P}_{ij} B_{i,3}(u) B_{j,3}(w), \quad (u,w) \in [0,1] \times [0,1] \tag{2.75}$$

which can also be written in a matrix form as

$$\mathbf{S}(u,w) = \mathbf{U}\mathbf{N}^{\mathrm{B}}\mathbf{G}^{\mathrm{B}}\mathbf{N}^{\mathrm{B}^{\mathrm{T}}}\mathbf{W}^{\mathrm{T}}, \quad (u,w) \in [0,1] \times [0,1] \tag{2.76}$$

where

$$\mathbf{N}^{\mathrm{B}} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.77}$$

which is identical to that of the cubic Bézier curve, and

$$\mathbf{G}^{\mathrm{B}} = \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{02} & \mathbf{P}_{03} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{12} & \mathbf{P}_{13} \\ \mathbf{P}_{20} & \mathbf{P}_{21} & \mathbf{P}_{22} & \mathbf{P}_{23} \\ \mathbf{P}_{30} & \mathbf{P}_{31} & \mathbf{P}_{32} & \mathbf{P}_{33} \end{bmatrix}_{4\times4\times3} \tag{2.78}$$

which consists of the 16 control points arranged in a 4×4 matrix form.

2.3.2 B-SPLINE SURFACE

Similar to a Bézier surface, a B-spline surface is defined by basis functions and the control polyhedron as

$$\mathbf{S}(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{P}_{ij} N_{i,k}(u) N_{j,\ell}(w), \quad (u,w) \in [0, n-k+2] \times [0, m-\ell+2] \tag{2.79}$$

where $N_{i,k}(u)$ and $N_{j,}\ell(w)$ are the same basis functions as those of the B-spline curves and $\mathbf{P}_{ij}$ is the control point of the ith row at the jth location in the $(n+1)\times(m+1)$ matrix. In Eq. 2.79, the polynomial orders of the basis functions $N_{i,k}(u)$ and $N_{j,\ell}(w)$ are $k-1$ and $\ell-1$, respectively. Note that k does not have to be equal to ℓ, implying that the polynomial orders of the B-spline surface along the u and w directions do not have to be identical.

Depending on the choice of the basis functions (e.g., uniform or nonuniform and polynomial orders), numerous types of surfaces can be adequately modeled using B-spline surfaces, as illustrated in Figure 2.20. The three surfaces in Figure 2.20(a) are open–open (i.e., open in both u and w

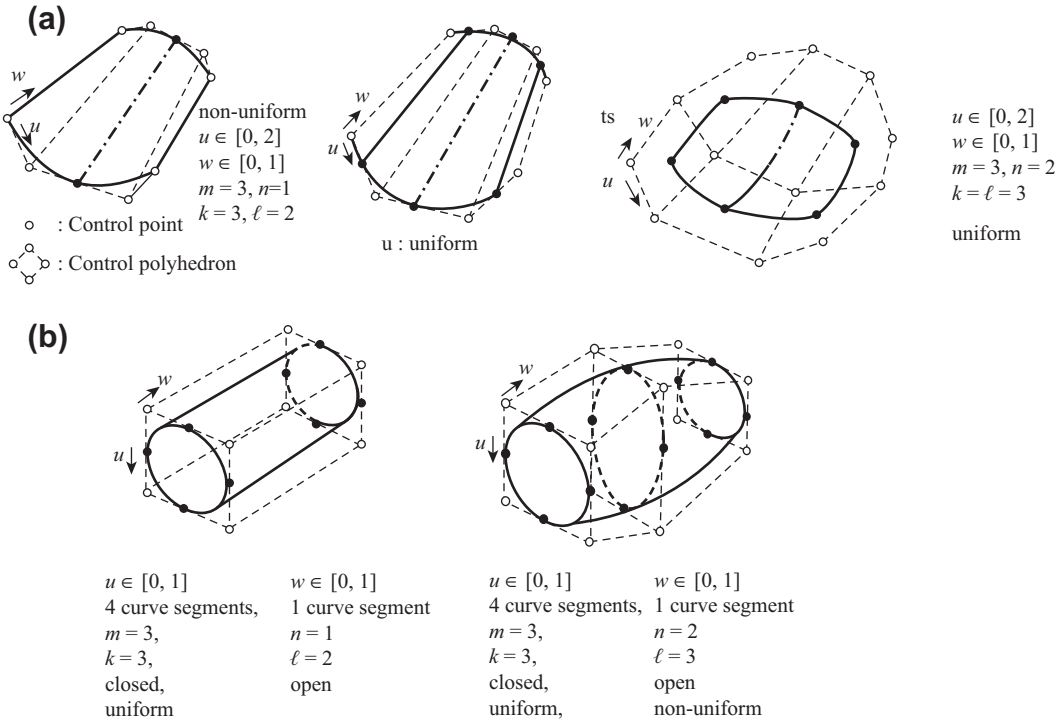


FIGURE 2.20

Various types of B-spline surfaces. (a) Open–open (left: nonuniform; middle: uniform in the u direction; right: uniform in both u and w directions). (b) Open–close: open in the w direction and closed in the u direction (left: linear in w direction, right: quadratic in w direction).

directions) and the two in Figure 2.20(b) are open–close (open in the w direction and closed in u direction). The surface on the left in Figure 2.20(a) employs nonuniform basis functions in the u direction, and the surface in the middle employs uniform basis functions. In both surfaces, a straight line is assumed in the w direction. The surface on the right assumes uniform basis functions in both the u and w directions. Both are quadratic. Both surfaces in Figure 2.20(b) assume a quadratic B-spline curve in the u direction. The surface on the left assumes a straight line along the w direction, and the one on the right employs nonuniform quadratic basis functions.

Similar to the B-spline surfaces, a NURB surface can be defined as

$$\mathrm{S}(u,w) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} \mathrm{h}_{ij}\mathbf{P}_{ij}\mathrm{N}_{i,k}(u)\mathbf{M}_{j,\ell}(w)}{\sum_{i=0}^{n}\sum_{j=0}^{m} \mathrm{h}_{ij}\mathrm{N}_{i,k}(u)\mathbf{M}_{j,\ell}(w)}, \quad (u,w) \in [0, n-k+2] \times [0, m-\ell+2]. \tag{2.80}$$

The major difference between a NURB and a B-spline surface is that the NURB surface is able to represent regular surfaces, such as a sphere, ellipsoid, and so on, just like that of NURB curves being able to represent Conic curves, such as circle or ellipse.

2.4 CAD-GENERATED SURFACES

With the knowledge of basic geometric modeling discussed in Sections 2.2 and 2.3, we are moving one step further to discuss surfaces generated by CAD. In CAD, we sketch an open profile and protrude it for a surface or protrude a closed profile for a solid feature. The protrusion capabilities commonly available in CAD include extrusion, blend (or loft), revolve, and sweep, as illustrated in Figure 2.21.

From a geometric modeling perspective, extruding a profile curve generates a cylindrical surface (Figure 2.21(a)). Sweeping a profile along a path curve leads to a sweep surface (Figure 2.21(b)). Revolving a sketch profile along an axis produces a surface of revolution (or revolved surface), as shown in Figure 2.21(c). Lofting two parallel sketch profiles without guide curves yields a ruled surface. Lofting more than two parallel sketch profiles (or two profiles with guide curves shown in Figure 2.21(d)) creates a loft surface.

In this section, we discuss mathematic representations for the parametric surfaces generated by the four types of protrusion discussed.

2.4.1 CYLINDRICAL SURFACES

As discussed earlier, in geometric modeling, a cylindrical surface can be considered as sweeping a straight line along a path curve $\mathbf{P}(u)$, as shown in Figure 2.22(a). Mathematically, such a surface can be written in a parametric form as

$$\mathbf{S}(u, w) = \mathbf{P}(u) + w\mathbf{r}, \quad (u, w) \in [0, 1] \times [0, 1] \tag{2.81}$$

in which u and w are the parametric coordinates of the surface, and $\mathbf{r}$ is the vector of the straight line. Note that the vector $\mathbf{r}$ can also be written as $\mathbf{r} = \mathbf{P}_2 - \mathbf{P}_0$, where $\mathbf{P}_0$ and $\mathbf{P}_2$ are the start and end points of the straight line, respectively. Note that the curve $\mathbf{P}(u)$ and the straight line are in space in general.

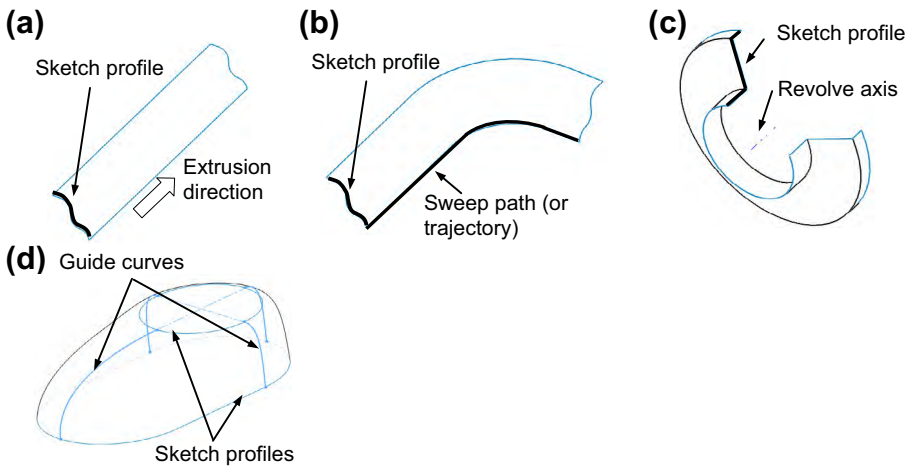


FIGURE 2.21

Protrusion of a profile for surface or solid. (a) Extrusion. (b) Sweep. (c) Revolve. (d) Loft (or blend).

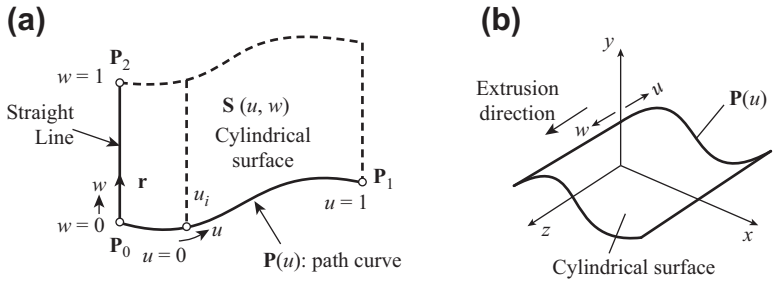


FIGURE 2.22

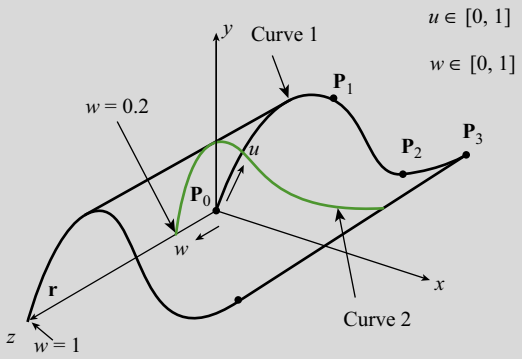
Cylindrical surface. (a) Sweeping straight line along a path curve $\mathbf{P}(u)$. (b) Extruding curve $\mathbf{P}(u)$ along a straight line perpendicular to the sketch curve $\mathbf{P}(u)$.

Certainly, a same cylindrical surface can be generated by extruding the curve $\mathbf{P}(u)$ along the straight line. If curve $\mathbf{P}(u)$ is part of a sketch profile in CAD, as shown in Figure 2.22(b), when the sketch is extruded, a cylindrical surface is generated, representing the boundary geometry of a solid feature. In this case, the straight line is always perpendicular to the sketch plane where the curve $\mathbf{P}(u)$ resides. The same equation in 2.81 represents the cylindrical surface.

The following example illustrates more details in constructing the mathematical representation for a cylindrical surface. We also include a Matlab script to graph the surface. Note that instead of using a Matlab surface graph function (e.g., surface(x,y,z)) to plot the surface, we use plot3(x,y,z) to plot points and line segments that show a surface with mesh. We hope such graphs offer you more insight into understanding the mathematic representation of parametric surfaces.

EXAMPLE 2.11

Find the parametric equation of the cylindrical surface generated by extruding a cubic spline curve on the x–y plane along the z-direction for 5 units, as shown below. Note that the four points that form the cubic spline curve are given as $\mathbf{P}_0 = [0,0,0]$, $\mathbf{P}_1 = [1,2.5,0]$, $\mathbf{P}_2 = [2,1,0]$, and $\mathbf{P}_3 = [4,2,0]$.



Continued

EXAMPLE 2.11—cont'd

Solutions

Using Eq. 2.34, the parametric equation for the curve $\mathbf{P}(u)$ can be written as

$$\mathbf{P}(u) = \mathbf{U}\mathbf{N}^{s}\mathbf{G}^{s} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -9/2 & 27/2 & -27/2 & 9/2 \\ 9 & -45/2 & 18 & -9/2 \\ -11/2 & 9 & -9/2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 1 & 2.5 & 0 \\ 2 & 1 & 0 \\ 4 & 2 & 0 \end{bmatrix}$$
$$= \left[4.5u^3 - 4.5u^2 + 4u, 29.5u^3 - 47.25u^2 + 20u, 0\right], \quad u \in \left[0, 1\right].$$

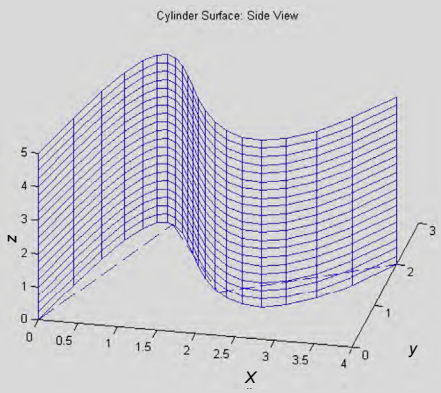
The extrusion vector $\mathbf{r}$ is

$$\mathbf{r} = [0,\ 0,\ 5]$$

Therefore, from Eq. 2.81, the parametric equation of the cylindrical surface is

$$\mathbf{S}(u, w) = \mathbf{P}(u) + \mathbf{r}w = \left[4.5u^3 - 4.5u^2 + 4u, 29.5u^3 - 47.25u^2 + 20u, 5w\right], \quad (u, w) \in \left[0, 1\right] \times \left[0, 1\right].$$

The surface is graphed in Matlab with the script shown below.



Matlab Script:

```
N=21;                                  % Create u value from 0 to 1 with increment
0.05 [1/(N-1)]
u=linspace(0,1,N);                     % Define vector u of N entries, with numbers
from 0 to 1
w= linspace(0,1,N);                    % Evenly spaced. Define vector w of N entries
Px=[0 1 2 4];                          % Control points (and next two lines)
Py=[0 2.5 1 2];
Pz=[0 0 0 0];
r=5;                                   % Define extrusion magnitude r=5
for i=1:1:N                            % Start a loop for fixing w value
x=4.5*u.^3-4.5*u.^2+4*u;
y=29.25*u.^3-47.25*u.^2+20*u;
z=(w(i)*r)*ones(1,N);                  % Define a vector of N entries with identical
value 1
figure(2);                             % Define a figure pointer for plot
plot3(x,y,z);                          % Make a 3D plot
hold on;                               % Hold the plot for additional data, curves, etc.
end;                                   % End the loop
for i=1:1:N                            % Start a loop for fixing u value
x=(4.5*u(i)^3-4.5*u(i)^2+4*u(i))*ones(1,N);
y=(29.25*u(i)^3-47.25*u(i)^2+20*u(i))*ones(1,N);
z=(w*r);
figure(2);
plot3(x,y,z);
hold on;
end;
plot3(Px,Py,Pz,'--');
xlabel('x'),ylabel('y'),zlabel('z');   % Add labels
title('Cylinder Surface: Front View'); % Add title
view(-55,-25);                         %View(azimuth, elevation)
title('Cylinder Surface: Back View');
view(55,-25);
title('Cylinder Surface: Side View');
hold off
```

2.4.2 RULED SURFACES

A ruled surface is defined by two path curves on the opposite sides of the surface, in which the trace of a straight line with its start and end points pass through the respective path curves with the same parametric value generates a ruled surface, as illustrated in Figure 2.23(a). The simplest of all ruled surfaces are plane, cone, and cylindrical surfaces. In addition, a surface with boundaries formed by four straight lines that are not coplanar is not a flat surface but a ruled surface, as shown in Figure 2.23(b). In this case, both path curves $\mathbf{P}(u)$ and $\mathbf{Q}(u)$ are straight lines, which are not necessarily coplanar.

Given two distinct curves $\mathbf{P}(u)$ and $\mathbf{Q}(u)$, as shown in Figure 2.24(a), a ruled surface is constructed by joining two points of the same u value (e.g., u^*) on the curves $\mathbf{P}(u)$ and $\mathbf{Q}(u)$, respectively, with a straight line; and sweeping the straight line along the two path curves with the same u value. Because the line connecting the two points of the same u value (such as u^*), respectively on curves $\mathbf{P}(u)$ and $\mathbf{Q}(u)$ is a straight line, this straight line can be written as

$$\mathbf{S}(u^*, w) = (1 - w)\mathbf{P}(u^*) + w\mathbf{Q}(u^*), \quad w \in [0, 1]. \tag{2.82a}$$

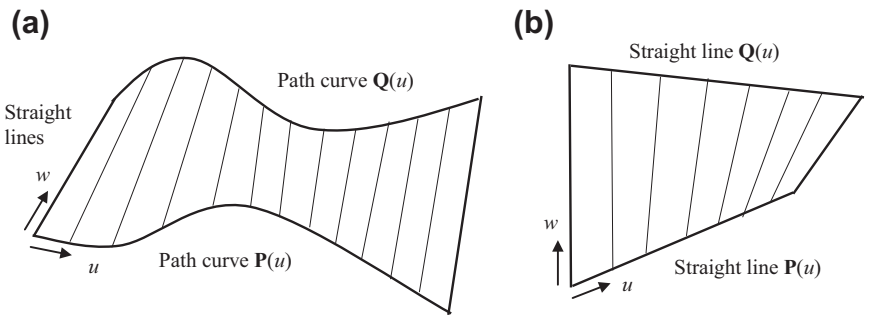


FIGURE 2.23

Ruled surfaces. (a) General ruled surface formed by two path curves $\mathbf{P}(u)$ and $\mathbf{Q}(u)$. (b) Ruled surface formed by two straight lines $\mathbf{P}(u)$ and $\mathbf{Q}(u)$ that are not coplanar.

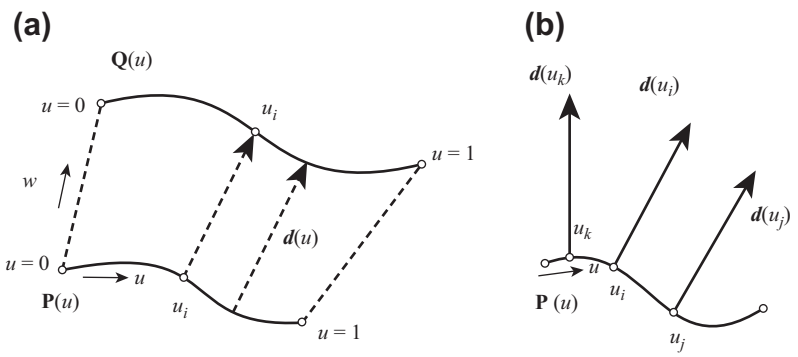


FIGURE 2.24

Ruled surfaces. (a) General ruled surface formed by two path curves $\mathbf{P}(u)$ and $\mathbf{Q}(u)$. (b) The same ruled surface generated by sweeping a non-constant vector $\mathbf{d}(u) = \mathbf{Q}(u) - \mathbf{P}(u)$ along the path curve $\mathbf{P}(u)$.

Because Eq. 2.82a is true for any u in [0,1], it can be generalized as a ruled surface by setting u ∈ [0,1] (i.e., removing the superscript * for the parameter u in Eq. 2.82a) as

$$\mathbf{S}(u,w) = (1-w)\mathbf{P}(u) + w\mathbf{Q}(u), \quad (u,w) \in [0,1] \times [0,1] \tag{2.82b}$$

which can also be rewritten as

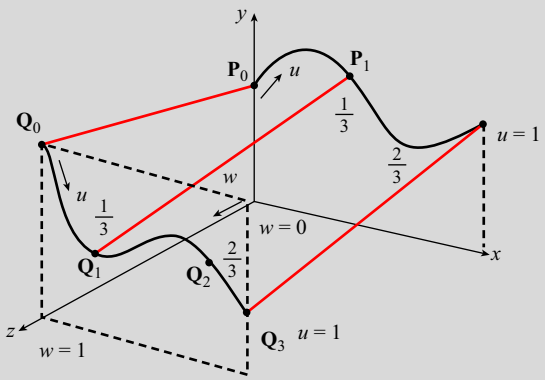
$$\mathbf{S}(u,w) = \mathbf{P}(u) + w(\mathbf{Q}(u) - \mathbf{P}(u)) = \mathbf{P}(u) + w\,\mathbf{d}(u), \ (u,w) \in [0,1] \times [0,1] \tag{2.82c}$$

which indicates that the same ruled surface can be generated by sweeping a nonconstant vector $\mathbf{d}(u) = \mathbf{Q}(u) - \mathbf{P}(u)$ along the path curve $\mathbf{P}(u)$, as shown in Figure 2.24(b).

The following example shows more details in constructing parametric equations for a ruled surface. Again, a Matlab script is included to graph the surface.

EXAMPLE 2.12

Find the parametric equation of the ruled surface generated by two path curves of cubic spline curves. Curve $\mathbf{P}(u)$ is the same as that of Example 2.11 and resides on the x–y plane. Curve $\mathbf{Q}(u)$ resides on a plane that is parallel to the x–y plane and offset 5 units along the z-direction, as shown below. Note that the four points that form the cubic spline curve $\mathbf{Q}(u)$ are given as $\mathbf{Q}_0 = [0,2,5]$, $\mathbf{Q}_1 = [1,1,5]$, $\mathbf{Q}_2 = [2,2.5,5]$, and $\mathbf{Q}_3 = [4,1,5]$.



Solutions

Using Eq. 2.34, the parametric equation for curve $\mathbf{Q}(u)$ can be written as

$$\mathbf{Q}(u) = \mathbf{U}\mathbf{N}^s\mathbf{G}^s = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -9/2 & 27/2 & -27/2 & 9/2 \\ 9 & -45/2 & 18 & -9/2 \\ -11/2 & 9 & -9/2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 2 & 5 \\ 1 & 1 & 5 \\ 2 & 2.5 & 5 \\ 4 & 1 & 5 \end{bmatrix}$$

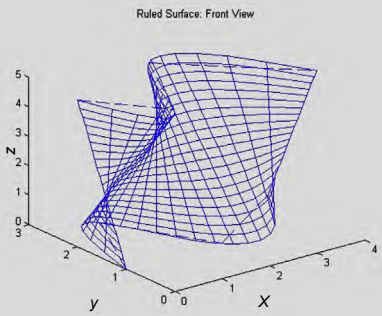
$$= \left[4.5u^3 - 4.5u^2 + 4u, 24.75u^3 + 36u^2 + 12.25u + 2.5, 0\right], \quad u \in [0,1].$$

EXAMPLE 2.12—cont'd

Therefore, from Eq. 2.82b, the parametric equation of the ruled surface is

$$\begin{aligned}\mathbf{S}(u,w) &= (1-w)\mathbf{P}(u) + w\mathbf{Q}(u) = (1-w)\left[4.5u^3 - 4.5u^2 + 4u, 29.5u^3 - 47.25u^2 + 20u, 0\right] \\ &\quad + w\left[4.5u^3 - 4.5u^2 + 4u, 24.75u^3 + 36u^2 + 12.25u + 2.5, 0\right], \quad (u,w) \in [0,1] \times [0,1].\end{aligned}$$

The surface is graphed in Matlab with the script shown below.




```
Matlab Script:

N=21
u=linspace(0,1,N);
w=linspace(0,1,N);

Gpx=[0 1 2 4];
Gpy=[1 2.5 1 2];
Gpz=[0 0 0 0];
Gqx=[0 1 2 4];
Gqy=[2 1 2.5 1];
Gqz=[5 5 5 5];

for i=1:1:N
Sx=(1-w(i))*(4.5*u.^3-4.5*u.^2+4*u)+w(i)*(4.5*u.^3-4.5*u.^2+4*u);
Sy=(1-w(i))*(24.75*u.^3-38.25*u.^2+14.5*u+1)+w(i)*(-24.75*u.^3+36*u.^2-12.25*u+2);
Sz=w(i)*5*ones(1,N);
plot3(Sx,Sy,Sz);
hold on;
end;

for i=1:1:N
Sx=(1-w)*(4.5*u(i)^3-4.5*u(i)^2+4*u(i))+w*(4.5*u(i)^3-4.5*u(i)^2+4*u(i));
Sy=(1-w)*(24.75*u(i)^3-38.25*u(i)^2+14.5*u(i)+1)+w*(-24.75*u(i)^3+36*u(i)^2-
12.25*u(i)+2);
Sz=w*5;
plot3(Sx,Sy,Sz);
hold on;
end;

plot3(Gpx,Gpy,Gpz,'--');
hold on;
plot3(Gqx,Gqy,Gqz,'--');

xlabel('x'),ylabel('y'),zlabel('z');
title('Ruled Surface: Front View');

hold off
```

2.4.3 LOFT (OR BLEND) SURFACES

In CAD, when we loft a solid or surface feature using more than two sketch profiles, we generate a loft (or blend) surface, instead of a ruled surface. For example, a loft surface can be constructed by lofting three curves $\mathbf{P}(u)$, $\mathbf{Q}(u)$, and $\mathbf{R}(u)$ on three respective parallel sketch planes along the w direction, as shown in Figure 2.25(a). If we assume that the curve $\mathbf{Q}(u)$ is located at $w = 1/2$ of the loft surface, then any curve along the w direction that is formed by a fixed u value at the three respective curves; for

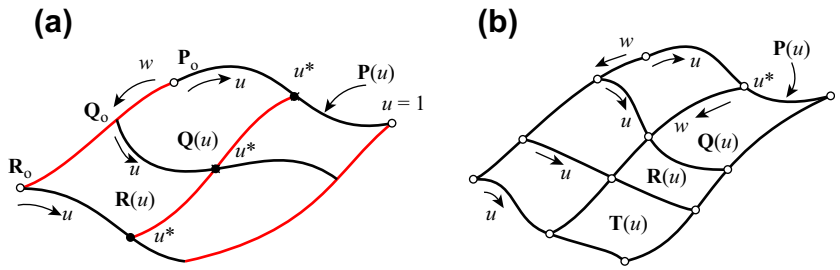


FIGURE 2.25

Loft surfaces. (a) Quadratic along the w direction by lofting three curves. (b) Cubic along the w direction by lofting four curves.

example, u^*, shown in Figure 2.25(a), is a quadratic spline curve formed by $\mathbf{P}(u^*)$, $\mathbf{Q}(u^*)$, and $\mathbf{R}(u^*)$, just like that of Eq. 2.14; i.e.,

$$\mathbf{S}(u^*, w) = \begin{bmatrix} w^2 & w & 1 \end{bmatrix} \begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}(u^*) \\ \mathbf{Q}(u^*) \\ \mathbf{R}(u^*) \end{bmatrix} = \mathrm{W}_{1\times 3}\mathrm{N}^{\mathrm{s}}_{3\times 3} \begin{bmatrix} \mathbf{P}(u^*) \\ \mathbf{Q}(u^*) \\ \mathbf{R}(u^*) \end{bmatrix}, \quad w \in [0, 1]. \tag{2.83}$$

Note that Eq. 2.83 is true for all u values in [0,1]; therefore, the parametric equation of the loft surface can be written as

$$\mathbf{S}(u, w) = \begin{bmatrix} w^2 & w & 1 \end{bmatrix} \begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}(u) \\ \mathbf{Q}(u) \\ \mathbf{R}(u) \end{bmatrix} = \mathrm{W}_{1\times 3}\mathbf{N}^{\mathrm{s}}_{3\times 3} \begin{bmatrix} \mathbf{P}(u) \\ \mathbf{Q}(u) \\ \mathbf{R}(u) \end{bmatrix}, \quad (u, w) \in [0, 1] \times [0, 1]. \tag{2.84}$$

Following the same fashion, a surface that lofts from four curves shown in Figure 2.25(b) can be written as follows, assuming that the four curves are located along the w direction of the loft surface at $w = 0$, 1/3, 2/3, and 1, respectively:

$$\mathbf{S}(u, w) = \begin{bmatrix} w^3 & w^2 & w & 1 \end{bmatrix} \begin{bmatrix} -9/2 & 27/2 & -27/2 & 9/2 \\ 9 & -45/2 & 18 & -9/2 \\ -11/2 & 9 & -9/2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}(u) \\ \mathbf{Q}(u) \\ \mathbf{R}(u) \\ \mathbf{T}(u) \end{bmatrix}$$

$$= \mathbf{W}_{1\times 4}\mathbf{N}^{\mathrm{s}}_{4\times 4} \begin{bmatrix} \mathbf{P}(u) \\ \mathbf{Q}(u) \\ \mathbf{R}(u) \\ \mathbf{T}(u) \end{bmatrix}, \quad (u, w) \in [0, 1] \times [0, 1]. \tag{2.85}$$

The following example shows more details in constructing parametric equations for a loft surface with three curves.

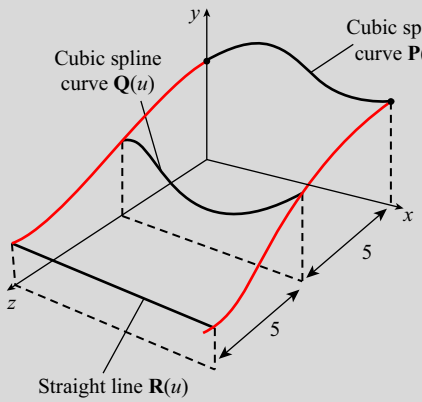
EXAMPLE 2.13

Find the parametric equation of the loft surface generated by lofting three curves on respective parallel planes (parallel to the x–y plane) with a uniformed space of 5 units, as shown below. The cubic spline curves $\mathbf{P}(u)$ and $\mathbf{Q}(u)$ are the same as those of Example 2.12. Curve $\mathbf{R}(u)$ is a straight line defined by $\mathbf{R}_0 = [0,1,10]$ and $\mathbf{R}_1 = [4,1,10]$.

Solutions

Using Eq. 2.10, the parametric equation for the straight line $\mathbf{R}$(u) can be written as

$$\mathbf{R}(u) = (1-u)\mathbf{R}_0 + u\mathbf{R}_0 = (1-u)[0,1,10] + u[4,1,10] = [4u,1,10].$$



Therefore, from Eq. 2.84, the parametric equation of the loft surface is

$$\mathbf{S}(u,w) = \begin{bmatrix} w^2 & w & 1 \end{bmatrix} \begin{bmatrix} 2 & -4 & 2 \\ -3 & 4 & -1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 4.5u^3 - 4.5u^2 + 4u, & 29.5u^3 - 47.25u^2 + 20u, & 0 \\ 4.5u^3 - 4.5u^2 + 4u, & 24.75u^3 + 36u^2 + 12.25u + 2.5, & 0 \\ 4u, & 0, & 10 \end{bmatrix}$$

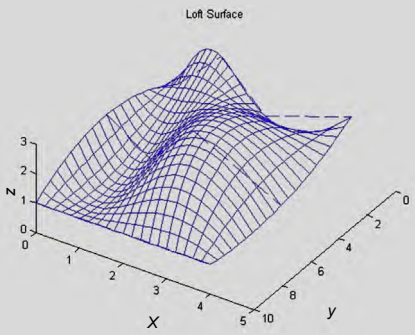
$$= \begin{bmatrix} 2w^2 - w + 1, & -4w^2 + 4w, & 2w^2 - w \end{bmatrix} \begin{bmatrix} 4.5u^3 - 4.5u^2 + 4u, & 29.5u^3 - 47.25u^2 + 20u, & 0 \\ 4.5u^3 - 4.5u^2 + 4u, & 24.75u^3 + 36u^2 + 12.25u + 2.5, & 0 \\ 4u, & 0, & 10 \end{bmatrix},$$

$$(u,w) \in [0,1] \times [0,1].$$

The surface is graphed in Matlab with the script shown below on the next page.

Continued

EXAMPLE 2.13—cont'd




```
Matlab Script:
N=21
u=linspace(0,1,N);
w=linspace(0,1,N);

Gpx=[0 1 2 4];
Gpz=[1 2.5 1 2];
Gpy=[0 0 0 0];
Gqx=[0 1 2 4];
Gqz=[2 1 2.5 1];
Gqy=[5 5 5 5];
Grx=[0 4];
Grz=[1 1];
Gry=[10 10];

for i=1:1:N
Sx=(2*w(i)^2-3*w(i)+1)*(4.5*u.^3-4.5*u.^2+4*u)+(-4*w(i)^2+4*w(i))*(4.5*u.^3-
4.5*u.^2+4*u)+(2*w(i)^2-w(i))*4*u;
Sz=(2*w(i)^2-3*w(i)+1)*(24.75*u.^3-38.25*u.^2+14.5*u+1)+(-4*w(i)^2+4*w(i))*(-
24.75*u.^3+36*u.^2-12.25*u+2)+(2*w(i)^2-w(i));
Sy=((-4*w(i)^2+4*w(i))*5+(2*w(i)^2-w(i))*10)*ones(1,N);
plot3(Sx,Sy,Sz);
hold on;
end;

for i=1:1:N
Sx=(2*w.^2-3*w+1)*(4.5*u(i)^3-4.5*u(i)^2+4*u(i))+(-4*w.^2+4*w)*(4.5*u(i)^3-
4.5*u(i)^2+4*u(i))+(2*w.^2-w)*4*u(i);
Sz=(2*w.^2-3*w+1)*(24.75*u(i)^3-38.25*u(i)^2+14.5*u(i)+1)+(-4*w.^2+4*w)*(-
24.75*u(i)^3+36*u(i)^2-12.25*u(i)+2)+(2*w.^2-w);
Sy=(-4*w.^2+4*w)*5+(2*w.^2-w)*10;
plot3(Sx,Sy,Sz);
hold on;
end;

plot3(Gpx,Gpy,Gpz,'--');
hold on;
plot3(Gqx,Gqy,Gqz,'--');
hold on;
plot3(Grx,Gry,Grz,'--');

xlabel('x'),ylabel('y'),zlabel('z');
title('Loft Surface');

view(-33,-58)
```

2.4.4 REVOLVED SURFACES

In CAD, when we sketch a profile and revolve it along an axis, the trace of the profile forms a revolved surface or surface of revolution. How do we represent the revolved surface in a parametric form? Consider the curve $\mathbf{P}(u)$ on the x–z plane, shown in Figure 2.26(a). If we revolve the curve along the z-axis counterclockwise for an angle of $\pi/2$ and pick just one point on the curve (e.g., $\mathbf{P}(u^*)$), to follow its trace, we will see that the trace of the point is a quarter circle (see Figure 2.26(b) in iso-view and viewed from the top, shown in Figure 2.26(c)) with center point O and radius $P_x(u^*)$. The quarter circle is located on a plane that is parallel with the x–y plane, but is elevated at a height $P_z(u^*)$, which is the

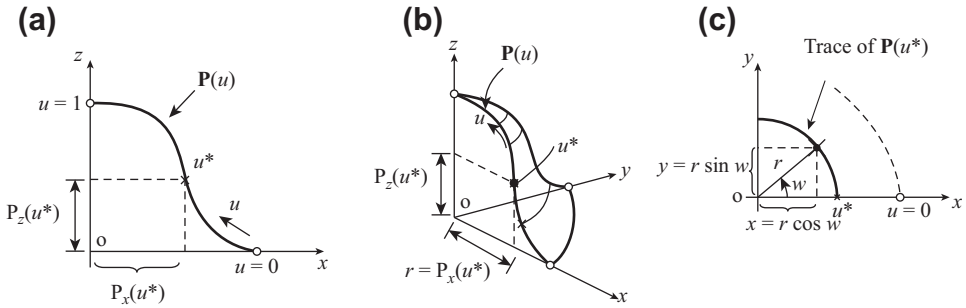


FIGURE 2.26

Surface of revolution. (a) Sketch profile $\mathbf{P}(u)$ in front view. (b) Revolved surface in iso-view. (c) Top view of the trace on the curve $\mathbf{P}(u^*)$.

z-component of the curve $\mathbf{P}(u)$ at $u = u^*$, as shown in Figures 2.26(a) and (b). Therefore, the parametric equation of the quarter circle can be written as

$$\mathbf{P}(w) = [\mathrm{P}_x(u^*)\cos w, \mathrm{P}_x(u^*)\sin w, \mathrm{P}_z(u^*)], w \in [0, \pi/2] \tag{2.86}$$

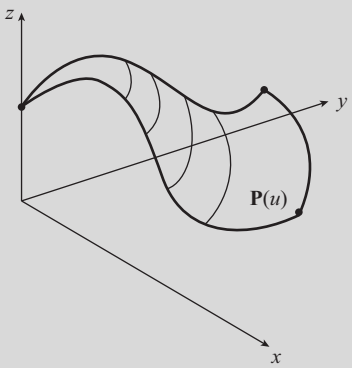
Because the above equation is true for any point u ∈ [0,1] on the curve $\mathbf{P}$(u), the trace of the curve creates a revolved surface written as

$$\mathbf{S}(u, w) = [\mathrm{P}_x(u)\cos w, \mathrm{P}_x(u)\sin w, \mathrm{P}_z(u)], u \in [0, 1], w \in [0, \pi/2]. \tag{2.87}$$

The following two examples show more details in constructing parametric equations for a revolved surface.

EXAMPLE 2.14

Find the parametric equation of the revolved surface generated by revolving a cubic spline curve $\mathbf{P}(u)$, which is identical to that of Example 2.11 on the x–z plane shown below with respect to the z-axis counterclockwise for a $\pi/2$ angle.



Continued

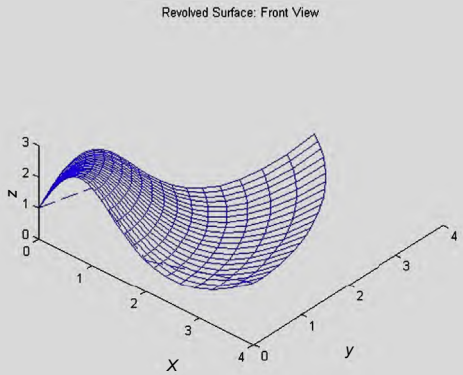
EXAMPLE 2.14—cont'd

Solutions

Using Eq. 2.87, the parametric equation for the revolved surface can be written as

$$\begin{aligned}\mathbf{S}(u,w) &= [\mathrm{P}_x(u)\cos w, \mathrm{P}_x(u)\sin w, \mathrm{P}_z(u)] \\ &= [(4.5u^3 - 4.5u^2 + 4u)\cos w, (4.5u^3 - 4.5u^2 + 4u)\sin w, 29.5u^3 - 47.25u^2 + 20u], u \in [0,1], w \in [0, \frac{\pi}{2}].\end{aligned}$$

The surface is graphed in Matlab with the script shown below.

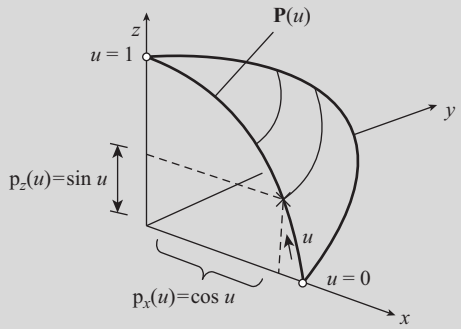


Matlab Script:

```
N=21
u=linspace(0,1,N);
w=linspace(0,1,N);

Gpx=[0 1 2 4];
Gpy=[0 0 0 0];
Gpz=[1 2.5 1 2];

for i=1:1:N
Sx=(4.5*u.^3-4.5*u.^2+4*u)*cos(w(i));
Sy=(4.5*u.^3-4.5*u.^2+4*u)*sin(w(i));
Sz=24.75*u.^3-38.25*u.^2+14.5*u+1;
plot3(Sx,Sy,Sz);
hold on;
end;

for i=1:1:N
Sx=(4.5*u(i)^3-4.5*u(i)^2+4*u(i))*cos(w);
Sy=(4.5*u(i)^3-4.5*u(i)^2+4*u(i))*sin(w);
Sz=(24.75*u(i)^3-38.25*u(i)^2+14.5*u(i)+1)*ones(1,N);
plot3(Sx,Sy,Sz);
hold on;
end;

plot3(Gpx,Gpy,Gpz,'--');
hold on;

xlabel('x'),ylabel('y'),zlabel('z');
title('Revolved Surface: Front View');

view(42,60);

hold off;
```

EXAMPLE 2.15

Find the parametric equation of the revolved surface generated by revolving the quarter circle of radius 1 on the x–z plane shown below with respect to the z axis for a $\pi/2$ angle.

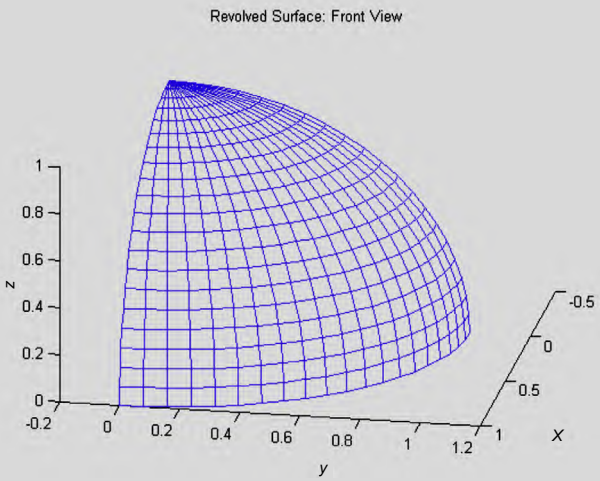


Solutions

Using Eq. 2.87, the parametric equation for the revolved surface can be written as

$$\begin{aligned} S(u,w) &= [P_x(u)\cos w, P_x(u)\sin w, P_z(u)] \\ &= [\cos u \ \cos w, \cos u \ \sin w, \sin u], u \in [0,1], w \in [0, \pi/2]. \end{aligned}$$

The surface is graphed in Matlab with the script shown below. Note that the circular arc $\mathbf{P}(u)$ can also be represented in a NURB form, such as by using the equation of Example 2.10. This is left as an exercise.



Matlab Script:

```
N=21
u=linspace(0,1.570796327,N);
w=linspace(0,1.570796327,N);

for i=1:1:N
Sx=cos(u)*cos(w(i));
Sy=cos(u)*sin(w(i));
Sz=sin(u);
plot3(Sx,Sy,Sz);
hold on;
end;

for i=1:1:N
Sx=cos(u(i))*cos(w);
Sy=cos(u(i))*sin(w);
Sz=sin(u(i))*ones(1,N);
plot3(Sx,Sy,Sz);
hold on;
end;

xlabel('x'),ylabel('y'),zlabel('z');
title('Revolved Surface: Front View');

view(100,30);

hold off
```

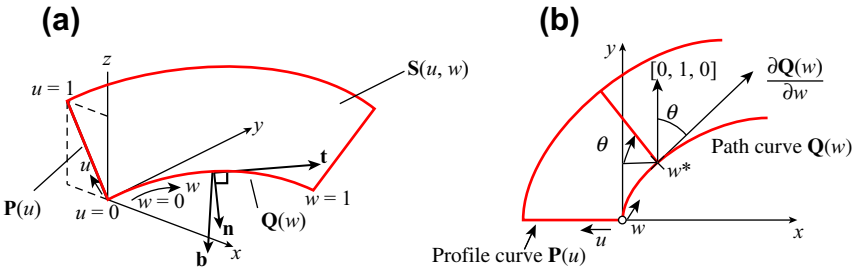


FIGURE 2.27

Sweep surface generated by sweeping profile curve $\mathbf{P}(u)$ along path curve $\mathbf{Q}(w)$. (a) The Frenet frame ($\boldsymbol{t}$, $\boldsymbol{b}$, $\boldsymbol{n}$). (b) Top view of the sweep surface.

2.4.5 SWEEP SURFACES

The trace of moving a profile curve $\mathbf{P}(u)$ along a path (or trajectory) curve $\mathbf{Q}(w)$ is a sweep surface $\mathbf{S}(u,w)$. For example, moving the straight line $\mathbf{P}(u)$ along the path curve $\mathbf{Q}(w)$ shown in Figure 2.27 generates a sweep surface $\mathbf{S}(u,w)$.

In CAD, we create a profile on a sketch plane and a path curve on a plane that is perpendicular with the profile sketch, and then we sweep the profile along the path to create a sweep solid feature or a sweep surface. If the path curve is a straight line, the sweep surface generated is nothing but a cylindrical surface. If the path curve is a circular arc, the resulting sweep surface is a surface of revolution. Therefore, both cylindrical and revolved surfaces can be considered as special cases of sweep surface. For sweeping a curve $\mathbf{P}(u)$ along a straight line, the orientation of the curve $\mathbf{P}(u)$ is not changing. While sweeping a curve $\mathbf{P}(u)$ along a circular arc, the curve orientation is constantly changing in order to ensure that the curve $\mathbf{P}(u)$ is always perpendicular to the path curve $\mathbf{Q}(w)$. If the path curve $\mathbf{Q}(w)$ is a general parametric curve, how can we orient the path curve $\mathbf{P}(u)$ properly so that it is always perpendicular to the path curve? This is an important characteristic of a sweep surface and requires our attention. The trick is attaching a smoothly varying coordinate system, a so-called Frenet frame, at any given location along the path curve $\mathbf{Q}(w)$.

A Frenet frame is defined by three independent direction vectors for a spatial curve. The vectors are (i) a normalized tangent vector, $\mathbf{t}(w)$, defined as

$$\mathbf{t}(w) = \text{normalized}\left(\frac{\partial \mathbf{Q}(w)}{\partial w}\right) = \frac{\mathbf{Q}_{,w}(w)}{\left\|\mathbf{Q}_{,w}(w)\right\|}; \tag{2.88a}$$

(ii) a normalized binormal vector, defined as

$$\mathbf{b}(w) = \text{normalized}\left(\mathbf{Q}_{,w}(w) \times \mathbf{Q}_{,ww}(w)\right) = \frac{\mathbf{Q}_{,w}(w) \times \mathbf{Q}_{,ww}(w)}{\left\|\mathbf{Q}_{,w}(w) \times \mathbf{Q}_{,ww}(w)\right\|}; \tag{2.88b}$$

and (iii) a normalized normal vector, defined as

$$\mathbf{n}(w) = \text{normalized}(\mathbf{b}(w) \times \mathbf{t}(w)) = \frac{\mathbf{b}(w) \times \mathbf{t}(w)}{\left\|\mathbf{b}(w) \times \mathbf{t}(w)\right\|}. \tag{2.88c}$$

The Frenet coordinate system (or frame) (**t**, **b**, **n**) varies smoothly, as we move along the path curve $\mathbf{Q}(w)$, as long as the curve is second-order differentiable; that is, $\mathbf{Q}_{,ww}(w)$ exists for all w. Using the smoothly varying Frenet frame, the trace of the profile curve $\mathbf{P}(u)$ at any given w value along the path curve (hence the sweep surface) can be determined by placing $\mathbf{P}(u)$ on the normal plane (spanned by vectors **b** and **n**), placing the start point of $\mathbf{P}(u)$ on the path curve $\mathbf{Q}(w)$, aligning $P_x(u)$ with vector **n**, and aligning $P_z(u)$ with vector **b**.

Mathematically, the parametric equation for a sweep surface, generated by sweeping a planar profile $\mathbf{P}(u)$ along a path curve $\mathbf{Q}(w)$ in space, can be defined as

$$\mathbf{S}(u,w) = \mathbf{Q}(w) + \mathbf{R}(w)(\mathrm{s}(w)\mathbf{P}(u)), \quad u \in [0,1] \;\; w \in [0,1] \tag{2.89}$$

where $\mathbf{R}(w)$ is a rotation matrix that rotates the profile curve $\mathbf{P}(u)$ so that its x- and z-components align with vectors **n** and **b**, respectively; and $\mathrm{s}(w)$ is a scale factor that scales the profile curve.

Note that in most sweep features in CAD, the scale factor is set to unity, and the path curve is usually a planar curve. For example, in Figure 2.27(a), the path curve $\mathbf{Q}(w)$ is sketched on the x–y plane. In this subsection, we assume that the path curve is placed on a plane that is perpendicular to that of the profile curve in order to simplify the mathematical equations of the sweep surface. Viewing from the top, as shown in Figure 2.27(b), it is apparent that in order to keep the profile curve $\mathbf{P}(u)$ perpendicular with the path curve $\mathbf{Q}(w)$ at any given w value (e.g., w^* in Figure 2.27(b)), the profile curve $\mathbf{P}(u)$ must rotate a θ angle clockwise along the z-axis. The θ angle can be calculated as

$$\theta(w) = \cos^{-1}\left(\begin{bmatrix}0 & 1 & 0\end{bmatrix} \cdot \mathbf{t}^{\mathrm{T}}(w)\right). \tag{2.90}$$

The rotation matrix is then obtained as

$$\mathbf{T}(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.91}$$

At the given value $w = w^*$, the profile curve $\mathbf{P}(u)$ is rotated clockwise with a θ angle clockwise along the z-axis, and then moved to the location of $\mathbf{Q}(w^*)$; that is,

$$\mathbf{S}(u,w^*) = \left[\mathbf{T}(\theta)\mathbf{P}(u)^{\mathrm{T}}\right]^{\mathrm{T}} + \mathbf{Q}(w^*), \quad u \in [0,1] \tag{2.92}$$

which is true for w $\in$ [0,1]. Hence, the parametric equation of the sweep surface can be written as

$$\mathbf{S}(u,w) = \left[\mathbf{T}(\theta)\mathbf{P}(u)^{\mathrm{T}}\right]^{\mathrm{T}} + \mathbf{Q}(w), \quad u \in [0,1], w \in [0,1]. \tag{2.93}$$

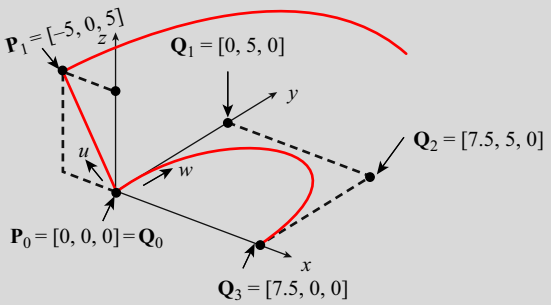
The following example shows more details in constructing parametric equations for a simple sweep surface.

EXAMPLE 2.16

Find the parametric equation of the sweep surface generated by sweeping a straight line along a cubic Bézier curve shown on the next page. The straight line is formed by connecting $\mathbf{P}_0 = [0,0,0]$ and $\mathbf{P}_1 = [-5,0,5]$; and the four control points of the cubic Bézier curve are $\mathbf{Q}_0 = [0,0,0]$, $\mathbf{Q}_1 = [0,5,0]$, $\mathbf{Q}_2 = [7.5,5,0]$, and $\mathbf{Q}_3 = [7.5,0,0]$.

Continued

EXAMPLE 2.16—cont'd



Solutions

From Eqs 2.10 and 2.42, the parametric equations of the straight line and the cubic Bézier curve can be written, respectively, as

$$\mathbf{P}(u) = (1-u)[0,0,0] + u[-5,0,5] = [-5u, 0, 5u]$$

and

$$\mathbf{Q}(w) = \begin{bmatrix} w^3 & w^2 & w & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{Q}_0 \\ \mathbf{Q}_1 \\ \mathbf{Q}_2 \\ \mathbf{Q}_3 \end{bmatrix} = \begin{bmatrix} w^3 & w^2 & w & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 0 \\ 7.5 & 5 & 0 \\ 7.5 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} -15w^3 + 22.5w^2, -15w^2 + 15w, 0 \end{bmatrix}.$$

The normalized tangent vector of the path curve $\mathbf{Q}(w)$ is

$$\mathbf{t}(w) = \frac{\mathbf{Q}_{,w}(w)}{\|\mathbf{Q}_{,w}(w)\|} = \frac{\begin{bmatrix} -45w^2 + 45w, 30w + 15, 0 \end{bmatrix}}{\sqrt{(-45w^2 + 45w)^2 + (30w + 15)^2 + 0^2}}$$

and the rotation angle θ can be obtained as

$$\theta = \cos^{-1}\left([0 \quad 1 \quad 0] \cdot \mathbf{t}^{\mathrm{T}}\right) = \cos^{-1}\left([0 \quad 1 \quad 0] \cdot \begin{bmatrix} \dfrac{-45w^2 + 45w}{\sqrt{(-45w^2 + 45w)^2 + (30w + 15)^2 + 0^2}} \\ \dfrac{-30w + 15}{\sqrt{(-45w^2 + 45w)^2 + (-30w + 15)^2 + 0^2}} \\ 0 \end{bmatrix} \right)$$

$$= \cos^{-1}\left(\frac{-30w + 15}{\sqrt{(-45w^2 + 45w)^2 + (-30w + 15)^2 + 0^2}} \right).$$

Now, rotating the curve $\mathbf{P}(u)$ an θ angle clockwise along the z-axis, we have

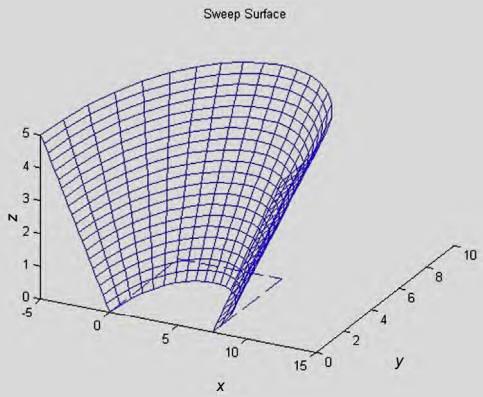
$$\mathbf{T}(\theta)\mathbf{P}(u)^{\mathrm{T}} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -5u \\ 0 \\ 5u \end{bmatrix} = \begin{bmatrix} -5u\cos\theta \\ 5u\sin\theta \\ 5u \end{bmatrix}.$$

EXAMPLE 2.16—cont'd

Hence, from Eq. 2.93, the sweep surface can be obtained as

$$\begin{aligned}\mathbf{S}(u,w) &= \left[\mathbf{T}(\theta)\mathbf{P}(u)^{\mathrm{T}}\right]^{\mathrm{T}} + \mathbf{Q}(w)\\ &= \left[-5u\cos\theta, 5u\sin\theta, 5u\right] + \left[-15w^3 + 22.5w^2, -15w^2 + 15, 0\right]\\ &= \left[-5u\cos\theta - 15w^3 + 22.5w^2, 5u\sin\theta - 15w^2 + 15, 5u\right], u \in [0,1], w \in [0,1].\end{aligned}$$

The surface is graphed in Matlab with the script shown below.



Matlab Script:

```
N=21
u=linspace(0,1,N);
w=linspace(0,1,N);

Px=[0 0 7.5 7.5];
Py=[0 5 5 0];
Pz=[0 0 0 0];

n=[0 1 0];

for i=1:1:N
tmag=((-45*w(i)^2+45*w(i))^2+(-30*w(i)+15)^2)^0.5;
t=[(-45*w(i)^2+45*w(i))/tmag (-30*w(i)+15)/tmag 0];
theta=acos(n*t');
Sx=-5*u.*cos(theta)-15*w(i)^3+22.5*w(i)^2;
Sy=5*u.*sin(theta)-15*w(i)^2+15*w(i);
Sz=5*u;
plot3(Sx,Sy,Sz);
hold on;
end;

tmag=((-45*w.^2+45*w).^2+(-30*w+15).^2).^0.5;
for i=1:1:N
t=[(-45*w(i)^2+45*w(i))./tmag(i) (-30*w(i)+15)./tmag(i) 0];
theta(i)=acos(n*t');
end;

for i=1:1:N
Sx=-5*u(i)*cos(theta)-15*w.^3+22.5*w.^2;
Sy=5*u(i)*sin(theta)-15*w.^2+15*w;
Sz=5*u(i)*ones(1,N);
plot3(Sx,Sy,Sz);
hold on;
end;

plot3(Px,Py,Pz,'--');
hold on;

xlabel('x'),ylabel('y'),zlabel('z');
title('Sweep Surface');

view(27,36);

hold off;
```

2.5 GEOMETRIC TRANSFORMATIONS

In geometric modeling, geometric entities, such as curves and surfaces, need to be constantly transformed for numerous purposes. The Euclidean transformations are the most commonly used transformations. A Euclidean transformation is a translation, a rotation, or a mirror. Euclidean transformations preserve length and angle measure. Moreover, the shape of a geometric entity will not change. That is, lines transform to lines, planes transform to planes, circles transform to circles, and ellipsoids transform to ellipsoids. Only the position and orientation of the object will change.

Another transformation, called affine transformation, is a generalization of Euclidean transformation. Under affine transformations, lines transform to lines; however, circles may become ellipses. Length and angle are not preserved. Essentially, an affine transformation is any transformation that preserves collinearity (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distances (e.g., the midpoint of a line segment remains the midpoint after transformation). Although an affine transformation preserves proportions on lines, it does not necessarily preserve angles or lengths. Geometric contraction, expansion, dilation, reflection, rotation, shear, similarity transformations, spiral similarities, and translation are all affine transformations, as are their combinations.

In this subsection, we discuss only the most basic transformations, including scaling, translation, and rotation. Note that by combining a number of transformations, a more sophisticated transformation, such as mirror or rotating along an arbitrarily axis, can be carried out.

Transformation of a parametric curve or surface can be accomplished by transforming its characteristic points, such as control points of Bézier or B-spline curves or surfaces, as well as tangent vectors (e.g., Hermit cubic curves), and twister vectors (e.g., the Coons patch). Mathematically, applying an affine (or Euclidean) transformation to a geometric entity, such as a B-spline curve $\mathbf{P}(u)$, can be expressed as

$$\mathbf{P}'(u) = \mathbf{T}\,\mathbf{P}(u) = \mathbf{T}\left(\sum_{i=0}^{n}\mathbf{P}_i \mathrm{N}_{i,k}(u)\right) = \sum_{i=0}^{n}(\mathbf{T}\mathbf{P}_i)\mathrm{N}_{i,k}(u) = \sum_{i=0}^{n}\mathbf{P}_i'\mathrm{N}_{i,k}(u) \tag{2.94}$$

where $\mathbf{T}$ is the affine transformation matrix, $\mathbf{P}_i'$ are the transformed characteristic points (in this case, control points), and the curve $\mathbf{P}'(u)$ is the transformed B-spline curve. In this section, we assume all curves and points are in column vector form.

Affine transformation is powerful and uniform mathematically. It is ideal for support of geometric transformations. To understand affine transformations, we need to first discuss homogeneous coordinates.

2.5.1 HOMOGENEOUS COORDINATES

Every point (x, y) in a 2D Cartesian plane has a corresponding set of homogeneous coordinates $(\mathrm{h}x, \mathrm{h}y, \mathrm{h})$ in the 3D projective space (also called the homogeneous space). When $\mathrm{h} = 1$, $(\mathrm{h}x, \mathrm{h}y, \mathrm{h})$ becomes $(x, y, 1)$, projecting $(\mathrm{h}x, \mathrm{h}y, \mathrm{h})$ point to the $\mathrm{h} = 1$ plane, as illustrated in Figure 2.28(a). Therefore, representing planar curves on a 2D Cartesian plane is a special case of the more general homogeneous coordinates.

Also, every point in the 3D Cartesian space (x, y, z) has a corresponding set of homogeneous coordinates $(\mathrm{h}x, \mathrm{h}y, \mathrm{h}z, \mathrm{h})$ in the four-dimensional (4D) projective space (again, called

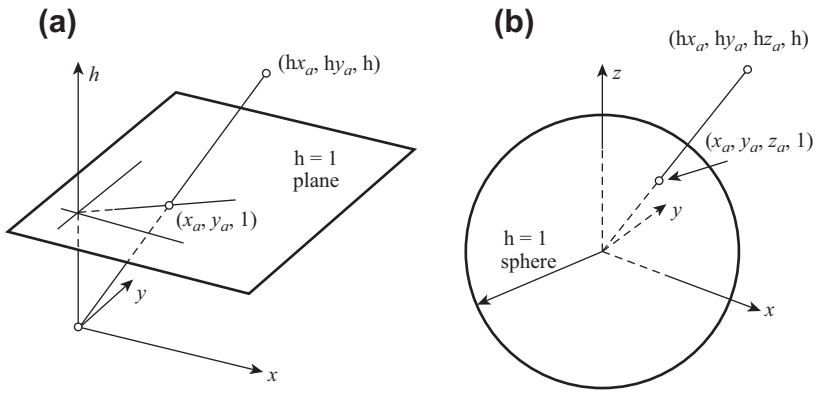


FIGURE 2.28

Homogeneous coordinates. (a) Two-dimensional. (b) Three-dimensional.

the homogeneous space). As illustrated in Figure 2.28(b), when h = 1, (hx, hy, hz, h) becomes (x, y, z, 1), which projects the point (hx, hy, hz, h) to the h = 1 sphere. Again, representing spatial curves and surfaces in 3D Cartesian space (h = 1) is a special case of the more general homogeneous coordinates.

Note that geometric transformations can be handled more effectively in the homogeneous coordinates than ordinary Cartesian coordinates, which is illustrated in the following example.

Let $\mathbf{P}_i$, $i = 0, n$, be $(n + 1)$ control points of a B-spline curve in the 4D homogeneous space with the same h; that is,

$$\mathbf{P}_i = [\,\mathrm{h}x_i\ \mathrm{h}y_i\ \mathrm{h}z_i\ \mathrm{h}\,]^{\mathrm{T}}_{1\times 4.} \tag{2.95}$$

Again, all points, such as $\mathbf{P}_i$, are represented in column vector form.

In homogeneous coordinates, the transformation matrix for an affine transformation can be defined by a 4×4 matrix as

$$\mathbf{T} = \begin{bmatrix} \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{M} \\ \mathrm{D} & \mathrm{E} & \mathrm{F} & \mathrm{N} \\ \mathrm{G} & \mathrm{H} & \mathrm{I} & \mathrm{O} \\ \mathrm{J} & \mathrm{K} & \mathrm{L} & \mathrm{S} \end{bmatrix} \tag{2.96}$$

in which the 3×3 matrix $\begin{bmatrix} \mathrm{A} & \mathrm{B} & \mathrm{C} \\ \mathrm{D} & \mathrm{E} & \mathrm{F} \\ \mathrm{G} & \mathrm{H} & \mathrm{I} \end{bmatrix}$ defines the scaling and rotation transformations, the 3×1 column vector $[\mathrm{M\ N\ O}]^{\mathrm{T}}$ determines the geometric translation, and the scalar [S] specifies the uniform global scaling. Note that the 1×3 row vector [J K L] is usually set to [0 0 0].

With this transformation matrix defined in Eq. 2.96, an affine transformation of points $\mathbf{P}_i$, $i = 0, n$, can be obtained as

$$\mathbf{P}'_i = \mathbf{T}\,\mathbf{P}_i, \text{ for } i = 0, n \tag{2.97}$$

where

$$\mathbf{P}_i' = [\,x_i' y_i' z_i' 1\,]^{\mathrm{T}} \text{is the point } \mathbf{P}_i = [\,x_i \mathrm{y}_i z_i 1\,]^{\mathrm{T}} \text{after transformation.}$$

2.5.2 SCALING

The transformation matrix for scaling a geometric entity is defined as

$$\mathbf{T}_{\mathrm{s}} = \begin{bmatrix} \mathrm{A} & 0 & 0 & 0 \\ 0 & \mathrm{E} & 0 & 0 \\ 0 & 0 & \mathrm{I} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.98}$$

where A, E, and I are the scaling factors for x, y, and z-coordinates, respectively, as illustrated in Figure 2.29(a). The rectangle of size $a \times b$ defined by four corner points $\mathbf{P}_0\ \mathbf{P}_1\ \mathbf{P}_2\ \mathbf{P}_3$ is scaled to be of size $\mathrm{A}a \times \mathrm{E}b$, defined by the transformed corners points $\mathbf{P}_0'\ \mathbf{P}_1'\ \mathbf{P}_2'\ \mathbf{P}_3'$, as shown below.

$$\begin{aligned} [\,\mathbf{P}_0' \quad \mathbf{P}_1' \quad \mathbf{P}_2' \quad \mathbf{P}_3'\,]_{\mathrm{s}} &= \mathbf{T}_{\mathrm{s}}[\,\mathbf{P}_0 \quad \mathbf{P}_1 \quad \mathbf{P}_2 \quad \mathbf{P}_3\,] = \begin{bmatrix} \mathrm{A} & 0 & 0 & 0 \\ 0 & \mathrm{E} & 0 & 0 \\ 0 & 0 & \mathrm{I} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathrm{P}_{0x} & \mathrm{P}_{1x} & \mathrm{P}_{2x} & \mathrm{P}_{3x} \\ \mathrm{P}_{0y} & \mathrm{P}_{1y} & \mathrm{P}_{2y} & \mathrm{P}_{3y} \\ \mathrm{P}_{0z} & \mathrm{P}_{1z} & \mathrm{P}_{2z} & \mathrm{P}_{3z} \\ 1 & 1 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{A} & 0 & 0 & 0 \\ 0 & \mathrm{E} & 0 & 0 \\ 0 & 0 & \mathrm{I} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & a & a & 0 \\ 0 & 0 & b & b \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & \mathrm{A}a & \mathrm{A}a & 0 \\ 0 & 0 & \mathrm{E}b & \mathrm{E}b \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{aligned} \tag{2.99}$$

Note that the four points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ can be control points of a parametric curve, such as a cubic Bézier curve. In this case, the same procedure shown above applies, as illustrated in Figure 2.29(b).

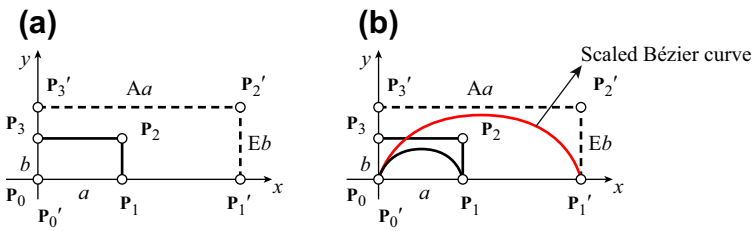


FIGURE 2.29

Scaling transformations. (a) Scaling a rectangle. (b) Scaling a cubic Bézier curve.

The 4×4 transformation matrix for a uniform global scaling can be defined as

$$\mathbf{T}_{\mathrm{s}}^{\mathrm{g}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathrm{S} \end{bmatrix} \tag{2.100}$$

where S is the scale factor. If we scale the rectangle shown in Figure 2.29(a) with a scale factor S, the rectangle of size $a \times b$ defined by four corner points $\mathbf{P}_0\ \mathbf{P}_1\ \mathbf{P}_2\ \mathbf{P}_3$ becomes $a/\mathrm{S} \times b/\mathrm{S}$, as shown below.

$$\begin{aligned} \begin{bmatrix} \mathbf{P}_0' & \mathbf{P}_1' & \mathbf{P}_2' & \mathbf{P}_3' \end{bmatrix}_{\mathrm{s}} &= \mathbf{T}_{\mathrm{s}}^{\mathrm{g}} \begin{bmatrix} \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathrm{S} \end{bmatrix} \begin{bmatrix} \mathrm{P}_{0x} & \mathrm{P}_{1x} & \mathrm{P}_{2x} & \mathrm{P}_{3x} \\ \mathrm{P}_{0y} & \mathrm{P}_{1y} & \mathrm{P}_{2y} & \mathrm{P}_{3y} \\ \mathrm{P}_{0z} & \mathrm{P}_{1z} & \mathrm{P}_{2z} & \mathrm{P}_{3z} \\ 1 & 1 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathrm{S} \end{bmatrix} \begin{bmatrix} 0 & a & a & 0 \\ 0 & 0 & b & b \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & a & a & 0 \\ 0 & 0 & b & b \\ 0 & 0 & 0 & 0 \\ \mathrm{S} & \mathrm{S} & \mathrm{S} & \mathrm{S} \end{bmatrix} \end{aligned} \tag{2.101}$$

Note that the matrix $\begin{bmatrix} 0 & a & a & 0 \\ 0 & 0 & b & b \\ 0 & 0 & 0 & 0 \\ \mathrm{S} & \mathrm{S} & \mathrm{S} & \mathrm{S} \end{bmatrix}$ is in the 4D homogeneous coordinates, which can be brought back to the Cartesian coordinates by dividing the entries by S; that is,

$$\begin{bmatrix} \mathbf{P}_0' & \mathbf{P}_1' & \mathbf{P}_2' & \mathbf{P}_3' \end{bmatrix}_{\mathrm{s}} = \begin{bmatrix} 0 & \frac{a}{\mathrm{S}} & \frac{a}{\mathrm{S}} & 0 \\ 0 & 0 & \frac{b}{\mathrm{S}} & \frac{b}{\mathrm{S}} \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{2.102}$$

2.5.3 TRANSLATION

The transformation matrix for translating a geometric entity is defined as

$$\mathbf{T}_{\mathrm{t}} = \begin{bmatrix} 1 & 0 & 0 & \mathrm{M} \\ 0 & 1 & 0 & \mathrm{N} \\ 0 & 0 & 1 & \mathrm{O} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.103}$$

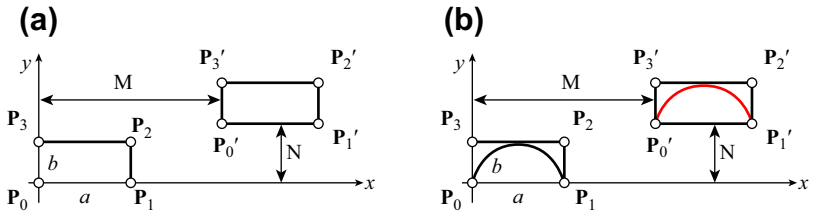


FIGURE 2.30

Geometry translation. (a) Translating a rectangle. (b) Translating a cubic Bézier curve.

where M, N, and O are the translation factors for the x-, y-, and z-coordinates, respectively, as illustrated in Figure 2.30(a). The rectangle of size $a \times b$ defined by four corner points $\mathbf{P}_0\ \mathbf{P}_1\ \mathbf{P}_2\ \mathbf{P}_3$ is translated to a new location, defined by the transformed corner points $\mathbf{P}_0'\ \mathbf{P}_1'\ \mathbf{P}_2'\ \mathbf{P}_3'$, as shown below.

$$\begin{aligned}\begin{bmatrix}\mathbf{P}_0' & \mathbf{P}_1' & \mathbf{P}_2' & \mathbf{P}_3'\end{bmatrix}_t &= \mathbf{T}_t\begin{bmatrix}\mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3\end{bmatrix} = \begin{bmatrix}1 & 0 & 0 & \mathrm{M}\\ 0 & 1 & 0 & \mathrm{N}\\ 0 & 0 & 1 & \mathrm{O}\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\mathrm{P}_{0x} & \mathrm{P}_{1x} & \mathrm{P}_{2x} & \mathrm{P}_{3x}\\ \mathrm{P}_{0y} & \mathrm{P}_{1y} & \mathrm{P}_{2y} & \mathrm{P}_{3y}\\ \mathrm{P}_{0z} & \mathrm{P}_{1z} & \mathrm{P}_{2z} & \mathrm{P}_{3z}\\ 1 & 1 & 1 & 1\end{bmatrix}\\ &= \begin{bmatrix}1 & 0 & 0 & \mathrm{M}\\ 0 & 1 & 0 & \mathrm{N}\\ 0 & 0 & 1 & \mathrm{O}\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0 & a & a & 0\\ 0 & 0 & b & b\\ 0 & 0 & 0 & 0\\ 1 & 1 & 1 & 1\end{bmatrix} = \begin{bmatrix}\mathrm{M} & a+\mathrm{M} & a+\mathrm{M} & \mathrm{M}\\ \mathrm{N} & \mathrm{N} & b+\mathrm{N} & b+\mathrm{N}\\ 0 & 0 & 0 & 0\\ 1 & 1 & 1 & 1\end{bmatrix}\end{aligned} \tag{2.104}$$

Note that, like before, the four points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$ can be control points of a parametric curve, such as a cubic Bézier curve. In this case, the same procedure shown above applies, as illustrated in Figure 2.30(b).

2.5.4 ROTATIONS

The transformation matrix for rotating a geometric entity on the x–y plane, such as a point $\mathbf{P}$ shown in Figure 2.31(a), along the z-axis at a positive angle θ can be written as

$$\mathbf{T}_{rz} = \begin{bmatrix}\cos\theta & -\sin\theta & 0 & 0\\ \sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}. \tag{2.105a}$$

Similarly, the matrices for rotating along the y- and x-axes, shown in Figures 2.31(b) and (c), respectively, can be written as

$$\mathbf{T}_{ry} = \begin{bmatrix}\cos\theta & 0 & -\sin\theta & 0\\ 0 & 1 & 0 & 0\\ \sin\theta & 0 & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{2.105b}$$

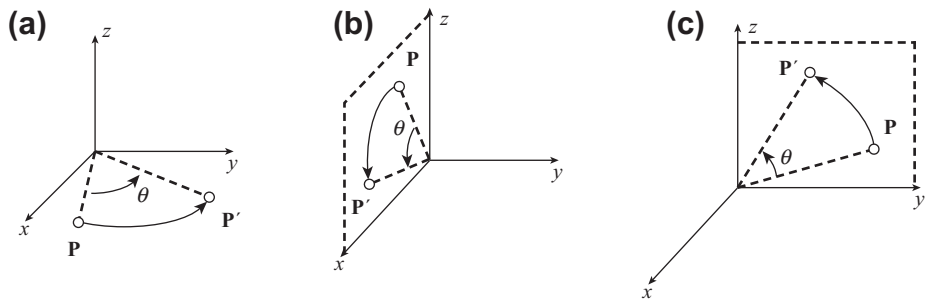


FIGURE 2.31

Rotation transformations. (a) Rotating a point along the *z*-axis. (b) Rotating a point along the *y*-axis. (c) Rotating a point along the *x*-axis.

and

$$\mathbf{T}_{rx} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.105c}$$

Is the order of rotation transformations interchangeable? For example, is $\mathbf{T}_{rx}(\alpha)\mathbf{T}_{ry}(\beta) = \mathbf{T}_{ry}(\beta)\mathbf{T}_{rx}(\alpha)$? The answer is generally no, unless the rotation angles α and β are infinitesimally small. Another important property worth mentioning is that these rotation transformation matrices shown in Eq. 2.105a–c are orthogonal; that is,

$$\mathbf{A}^{\mathbf{T}}\mathbf{A} = \mathbf{I} \tag{2.106a}$$

where matrix $\mathbf{A}$ is the rotation part of the transformation matrix; that is,

$$\mathbf{T}_{\mathrm{r}} = \begin{bmatrix} & & & 0 \\ & \mathbf{A} & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

In addition, the determinant of the matrix $\mathbf{A}$ is 1; that is,

$$|\mathbf{A}| = 1 \tag{2.106b}$$

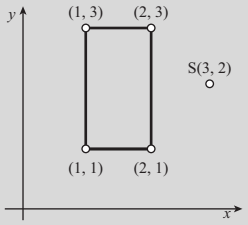
2.5.5 COMPOSITE TRANSFORMATIONS

On many occasions, a geometric transformation is accomplished by multiple transformations. For example, rotating a rectangle shown in the figure of Example 2.17 at a point other than the origin of the Cartesian coordinate system requires first translating the entity to a location where the rotating point coincides with the origin of the coordinate system. After rotating the entity anchored at the origin, the rotated entity must be translated back to its original location. Such a transformation is called a composite transformation. The 4×4 transformation matrix $\mathbf{T}_{\mathrm{c}}$ for a composite transformation consists

of multiplications of individual transform matrices in a prescribed order. In general, the order is not interchangeable.

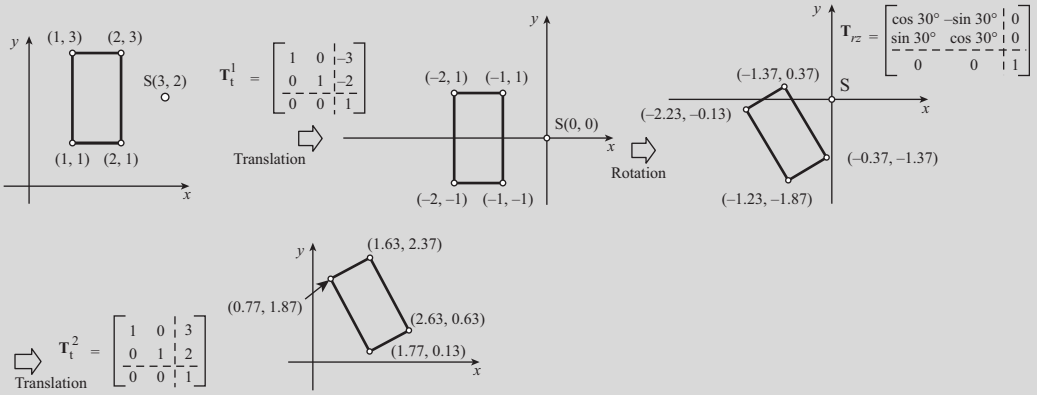
EXAMPLE 2.17

Find the composite transformation matrix that rotates the rectangle shown below at the point $\mathbf{S} = [3,2]^T$, a 30° angle. Note that the corner points of the rectangle are $\mathbf{P}_1 = [1,1]^T$, $\mathbf{P}_2 = [2,1]^T$, $\mathbf{P}_3 = [2,3]^T$ and $\mathbf{P}_4 = [1,3]^T$.



Solutions

There are three individual transformations involved. They are translating (−3,−2), rotating a 30° angle along the z-axis, and translating (3,2), as shown below. Note that because the transformation takes place on the x−y plane, we omit entities relevant to the z-component in the transformation matrices.



The individual transformation matrices are defined as

$$\mathbf{T}_t^1 = \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_{rz} = \begin{bmatrix} \cos 30° & -\sin 30° & 0 \\ \sin 30° & \cos 30° & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_t^2 = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix}.$$

Therefore, the composite transformation matrix can be calculated as

$$\mathbf{T}_c = \mathbf{T}_t^2 \mathbf{T}_{rz} \mathbf{T}_t^1 = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 30° & -\sin 30° & 0 \\ \sin 30° & \cos 30° & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.866 & -0.5 & 1.40 \\ 0.5 & 0.866 & -1.23 \\ 0 & 0 & 1 \end{bmatrix}.$$

Hence, the transformed rectangle is defined by the four transformed corner points as

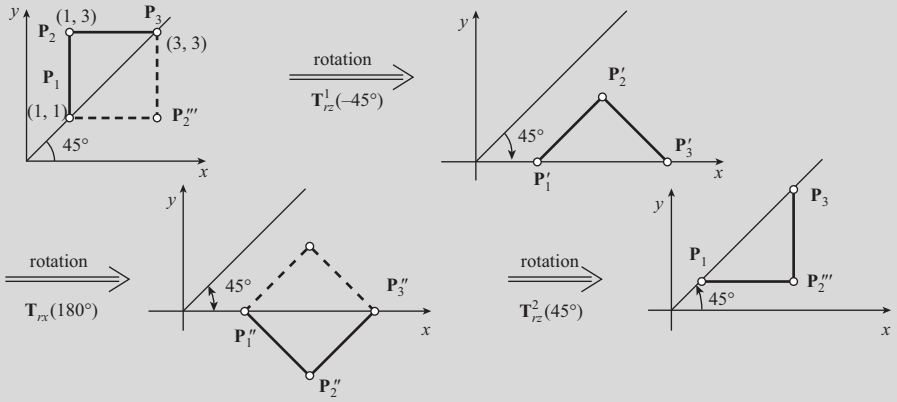
$$\mathbf{P}' = \mathbf{T}_c \mathbf{P} = \begin{bmatrix} 0.866 & -0.5 & 1.40 \\ 0.5 & 0.866 & -1.23 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 & 1 \\ 1 & 1 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1.77 & 2.63 & 1.63 & 0.77 \\ 0.13 & 0.63 & 2.37 & 1.87 \\ 1 & 1 & 1 & 1 \end{bmatrix}.$$

EXAMPLE 2.18

Mirror the isosceles triangle shown below along a 45°-axis. Note that the corner points of the triangle are $\mathbf{P}_1 = [1,1]^T$, $\mathbf{P}_2 = [1,3]^T$, and $\mathbf{P}_3 = [3,3]^T$.

Solutions

This mirror transformation can be accomplished by first rotating the triangle 45° clockwise along the z-axis so that its hypotenuse aligns with the x-axis. The triangle is then rotated along the x-axis by a 180° angle. Then, it is rotated 45° counterclockwise along the z-axis.



The composite transformation matrix, consisting of three individual rotation matrices, can be found as

$$\mathbf{T}_c = \mathbf{T}_{rz}^2 \mathbf{T}_{rx} \mathbf{T}_{rz}^1 = \begin{bmatrix} \cos(45°) & -\sin(45°) & 0 & 0 \\ \sin(45°) & \cos(45°) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(180°) & -\sin(180°) & 0 \\ 0 & \sin(180°) & \cos(180°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\times \begin{bmatrix} \cos(-45°) & -\sin(-45°) & 0 & 0 \\ \sin(-45°) & \cos(-45°) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Continued

EXAMPLE 2.18—cont'd

Hence, the transformed triangle is defined by the three transformed corner points as

$$\mathbf{P}' = \mathbf{T}_c\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 3 \\ 1 & 3 & 3 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 3 \\ 1 & 1 & 3 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

Is this the only way to perform the mirror transformation? The answer is no. You may try another composite transformation to mirror the triangle.

2.6 CASE STUDIES

Two case studies are included in this section. They are the curve fitting and surface skinning techniques, and applications of the techniques to engineering applications. We include four examples to demonstrate the modeling technique, including integration of topology and shape optimization, human middle ear, human tooth, and reverse engineering of an airplane tubing.

2.6.1 CURVE FITTING AND SURFACE SKINNING

In many engineering applications, discrete points extracted from a physical object often serve as a starting point for geometric model construction. One example is tracing the histological sections of a biological object. Through tracing outlines of the sections, discrete points are obtained and are employed to construct B-spline curves that represent the exterior contours of the components using a curve fitting technique. The surface skinning technique is then employed to quilt the B-spline curves for smooth boundary surfaces of the object.

2.6.1.1 Curve Fitting

The curve fitting technique employs the least square fitting for discrete points measured on a preselected section of an object. The best fitting curve can be obtained by minimizing the sum of the distance between the curve and the geometric points. Mathematically, the distance sum f is defined as

$$f = \sum_{j=0}^{r} \left\| \mathbf{P}_j - \mathbf{x}(u_j) \right\|^2 \tag{2.107}$$

where $\mathbf{P}_j$ is the position vector of the jth discrete point, and $r+1$ is the total number of points captured in the section contour; $||\cdot||$ is the norm of the vector $\cdot$, $\mathbf{x}(u)$ is the fitting B-spline curve, and $\mathbf{x}(u_j) = [x_1(u_j), x_2(u_j), x_3(u_j)]$ is the position vector of the fitting B-spline curve at u_j, where u is the parametric coordinate of the curve. The u_j in Eq. 2.107 is defined by the length ratio of the polygon formed by the geometric points $\mathbf{P}_j$, as illustrated in Figure 2.32(a). Mathematically, the values of u_j can be calculated by

$$u_0 = 0, \quad u_j = (r+1) \sum_{k=0}^{j-1} \left| \mathbf{P}_{(k+1)\bmod(r+1)} - \mathbf{P}_k \right| \Big/ \sum_{k=0}^{r} \left| \mathbf{P}_{(k+1)\bmod(r+1)} - \mathbf{P}_k \right|, \quad (j = 1, r). \tag{2.108}$$

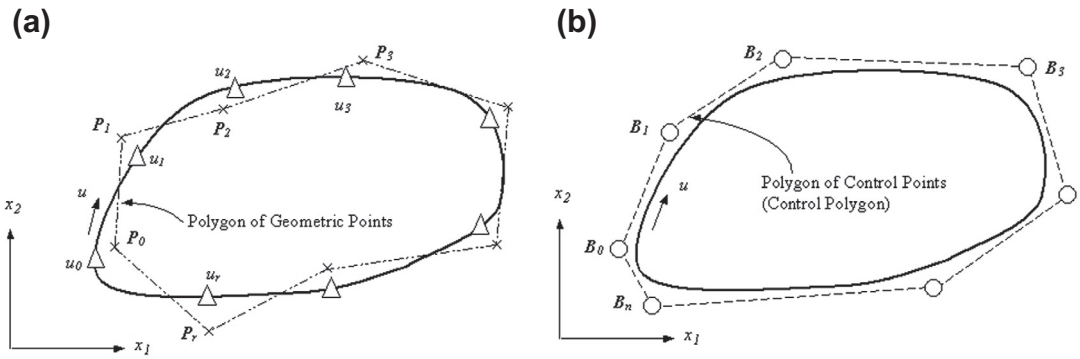


FIGURE 2.32

B-spline curve fitting. (a) Curve fitting for geometric points $\mathbf{P}_j$. (b) B-spline curve with control points $\mathbf{B}_i$.

The B-spline curve is defined as

$$\mathbf{x}(u) = \sum_{i=0}^{n} \mathbf{B}_i \mathrm{N}_{i,k}(u) \tag{2.109}$$

where $\mathbf{B}_\mathrm{i}$ is the ith control point shown in Figure 2.32(b), $n+1$ is the number of control points, and $\mathrm{N}_{i,k}(u)$ is the basis function of the B-spline curve, defined recursively as

$$\mathrm{N}_{i,k}(u) = \frac{(u - \mathrm{t}_i)\mathrm{N}_{i,k-1}(u)}{\mathrm{t}_{i+k-1} - \mathrm{t}_i} + \frac{(\mathrm{t}_{i+k} - u)\mathrm{N}_{i+1,k-1}(u)}{\mathrm{t}_{i+k} - \mathrm{t}_{i+1}} \quad \text{and} \quad \begin{cases} \mathrm{N}_{i,1}(u) = 1, & \text{if } \mathrm{t}_i \le u \le \mathrm{t}_{i+1} \\ \mathrm{N}_{i,1}(u) = 0, & \text{otherwise} \end{cases} \tag{2.110}$$

where $[\mathrm{t}_i, \mathrm{t}_{i+1}]$ is a knot span formed by the two consecutive knots t_i and t_{i+1}, and $k - 1$ is the polynomial order of the basis functions.

To minimize f, the derivatives of f with respect to the $n + 1$ control points are set to zero. For simplicity, considering only the ℓ^{th} control point, one has

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{B}_\ell} = \sum_{j=0}^{r} \left\| -2\mathbf{P}_j \sum_{i=0}^{n} \mathrm{N}_{i,k}(u_j) + 2\sum_{i=0}^{n} \mathrm{N}_{i,k}(u_j) \left(\sum_{i=0}^{n} \mathrm{N}_{i,k}(u_j)\mathbf{B}_\ell \right) \right\| = 0. \tag{2.111}$$

For $\ell = 0, n$, the above expression can be rewritten in a matrix form as

$$\mathbf{N}^{\mathrm{T}}\mathbf{N}\mathbf{B} = \mathbf{N}^{\mathrm{T}}\mathbf{P} \tag{2.112}$$

where $\mathbf{N} \in \mathbf{R}^{(r+1)\times(n+1)}$, $\mathbf{B} = \mathbf{R}^{(n+1)\times 3}$, $\mathbf{P} = \mathbf{R}^{(r+1)\times 3}$, and

$$\mathbf{N} = \begin{bmatrix} \mathrm{N}_{0,k}(u_0) & \mathrm{N}_{1,k}(u_0) & \cdots & \mathrm{N}_{n,k}(u_0) \\ \mathrm{N}_{0,k}(u_1) & \mathrm{N}_{1,k}(u_1) & \cdots & \mathrm{N}_{n,k}(u_1) \\ \vdots & \ddots & \ddots & \vdots \\ \mathrm{N}_{0,k}(u_\mathrm{r}) & \mathrm{N}_{1,k}(u_\mathrm{r}) & \cdots & \mathrm{N}_{n,k}(u_\mathrm{r}) \end{bmatrix}_{(r+1)\times(n+1)} \tag{2.113}$$

Note that $\mathbf{N}^{\mathrm{T}}\mathbf{N}$ is invertible if $\mathrm{N}_{i,k}(u_j) \neq 0$. This is true if and only if $\mathrm{t}_{i-k+1} < u_j < \mathrm{t}_{i+1}$, for $i = 0,n$, and $j = 0,r$. This implies that there must exist at least one u_j in at least one knot span so that $\mathrm{N}_{i,k}(u_j) \neq 0$ for all basis functions. This requirement can be achieved by adjusting the knot values of the basis functions. The curve fitting error can be controlled by adjusting the polynomial order and the number of control points. The output of the curve fitting is a set of control points and basis functions that describe the smoothed section contour.

2.6.1.2 Surface Skinning

The fitting B-spline curves discussed above are then "quilted" across sections to form an open B-spline surface, as shown in Figure 2.33, using the surface skinning technique. Note that in this process, the number of control points of the B-spline curves must be kept identical across sections. In addition, the polynomial order of the basis functions and knot values of the B-spline curves must be identical in all sections. The control points are connected to their corresponding points across sections, as shown in Figure 2.33(a), to form a control polyhedron. The enclosed B-spline surface is then constructed, as shown in Figure 2.33(b), by

$$\mathbf{S}(u,w) = \sum_{i=0}^{n}\sum_{j=0}^{m}\mathbf{B}_{ij}\mathrm{N}_{i,k}(u)\mathrm{N}_{j,\ell}(w), \tag{2.114}$$

where $n + 1$ and $m + 1$ are the numbers of control points in the u- and w-parametric directions, respectively; and $k - 1$ and $\ell - 1$ are the polynomial orders of the basis functions $\mathrm{N}_{i,k}(u)$ and $\mathrm{N}_{j,\ell}(w)$,

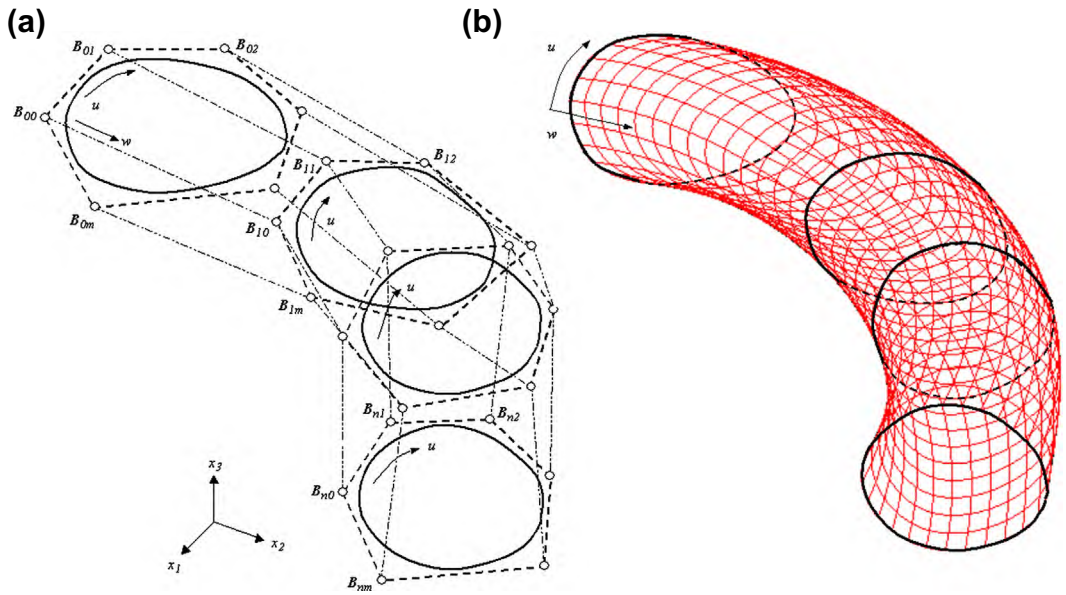


FIGURE 2.33

B-spline surface skinning. (a) Control polyhedron and section curves. (b) B-spline surface enclosed by the control polyhedron.

respectively. Note that the B-spline surface constructed is C^2-continuous in both u- and w-parametric directions, if cubic basis functions are assumed. The control points and basis functions of the B-spline surface can be imported into CAD software to support solid modeling.

2.6.2 ENGINEERING APPLICATIONS

The first application is the integration of topology and shape optimization. Topology optimization (Bendsoe and Sigmund, 2003) has drawn significant attention in the recent development of structural optimization. This method has been proven effective in determining the initial geometric shape for structural designs. The main drawback of the method, however, is that the topology optimization always leads to a nonsmooth structural geometry, while most of the engineering applications require a smooth geometric shape, especially for manufacturing. On the other hand, shape optimization (Chang and Choi, 1992) starts with a smooth geometric model that can be manufactured much easier. However, the optimal shape is confined to the topology of the initial structural geometry. No additional holes can be created during the shape optimization process. It is desirable to combine topology and shape optimizations to support structural design effectively by taking advantage of both methods. The curve fitting and surface skinning technique discussed in Section 2.6.1 is ideal to support integration of topology optimization and shape optimization.

To demonstrate the technique, a tracked vehicle roadarm shown in Figure 2.34(a) is optimized using topology optimization from the initial shape shown in Figure 2.34(b) to that of Figure 2.34(c) (Tang and Chang, 2001).

The optimal design is unsmooth and cannot be mass produced. Geometric points of five representative sections (Step 2 in Figure 2.35) of the roadarm are selected and fitted with B-spline curves (Steps 3a and 3b in Figure 2.35). Following the surface skinning method, an outer polyhedron formed by the 6×5 control points and the enclosed B-spline surface are created (Step 4a). Similarly, an inner B-spline surface (4×3 control points) that represents the hole in the roadarm is created (Step 4b). These B-spline surfaces are imported into SolidWorks for solid model construction. In SolidWorks, the outer and inner solid models are created by filling up the cavities enclosed by the outer and inner B-spline surfaces, respectively. The final solid model is obtained by subtracting the inner solid from the outer one (Step 5) and uniting the subtracted solid model with two end half cylinders, as shown in Figure 2.35.

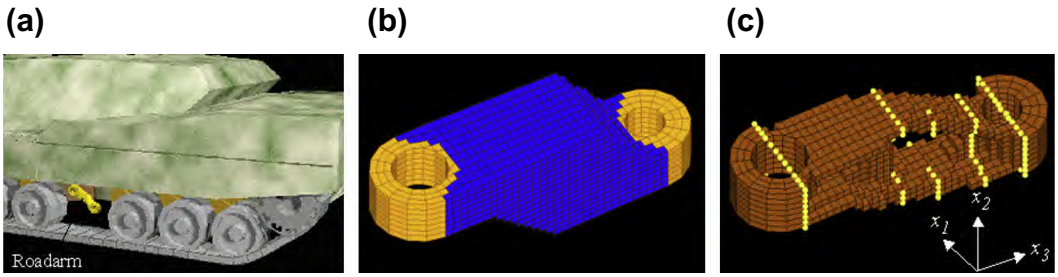


FIGURE 2.34

The tracked vehicle roadarm example. (a) Physical model. (b) Initial finite element model. (c) Topologically optimized model.

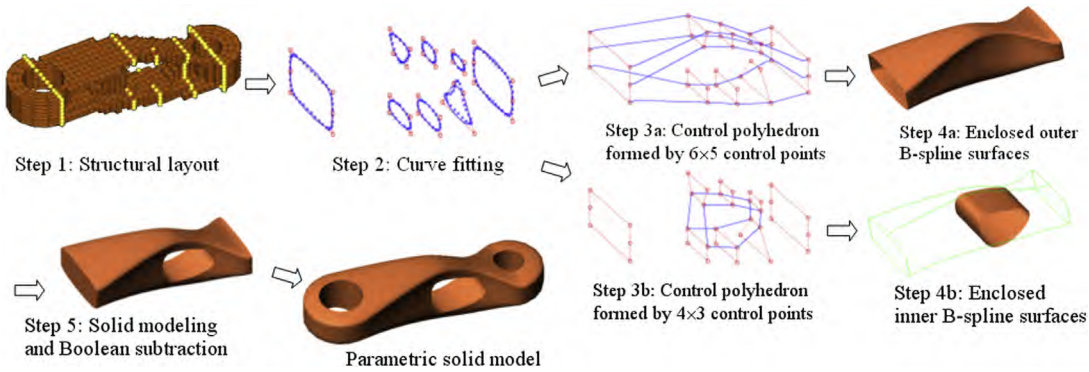


FIGURE 2.35

Construction of B-spline surfaces for structural design.

The second example is modeling a human middle ear (Sun et al., 2002). The modeling steps start with the histological section preparation of human temporal bone (Figures 2.36(a) and (b)). Through tracing outlines of the middle ear components on the sections (Figure 2.36(c)), a set of discrete points is obtained and employed to construct B-spline curves that represent the exterior contours of the components using the curve fitting technique (Figure 2.36(d)). The surface skinning technique is then employed to quilt the B-spline curves for smooth boundary surfaces of the middle ear components using B-spline surfaces (Figure 2.36(e)). The solid models of the middle ear components are constructed using these surfaces and then assembled to create a complete middle ear in CAD. The geometric model constructed using the proposed method is smooth and can be used to create finite element models for mechanics study (Figure 2.36(f)).

The same modeling technique is applied to a human maxillary second molar, which is the third example to be presented. The main purpose of constructing a geometric model for the human tooth is to capture accurately the geometry of the critical dentino–enamel junction (DEJ), which is important for investigating stress distribution inside the tooth. The geometric modeling started with a histological section preparation of a human tooth (Figure 2.37(a)). Through tracing outlines of the tooth on the sections, discrete points are obtained and are employed to construct B-spline curves that represent the exterior contours and DEJ of the tooth using the least square curve fitting technique (Figure 2.37(b)). The surface skinning technique is then employed to quilt the B-spline curves to create a smooth boundary and DEJ of the tooth using B-spline surfaces (Figure 2.37(c)). These surfaces are respectively imported into SolidWorks via its Application Protocol Interface (API) to create solid models, as shown in Figure 2.37(d) (Chang et al., 2003).

The last example is for support of reverse engineering. An airplane tubing sample part was first scanned using an industrial CT scanner, capturing both the interior and exterior geometry with 486,107 uniformly spaced data points (Figure 2.38). A B-spline curve fitting and surface skinning approach was employed to convert the data points into B-spline surfaces (Chang et al., 2006). A physical model was produced using a stereolithography apparatus (SLA) and mounted to the production fixtures to verify the accuracy of the surface model, as shown in Figure 2.38.

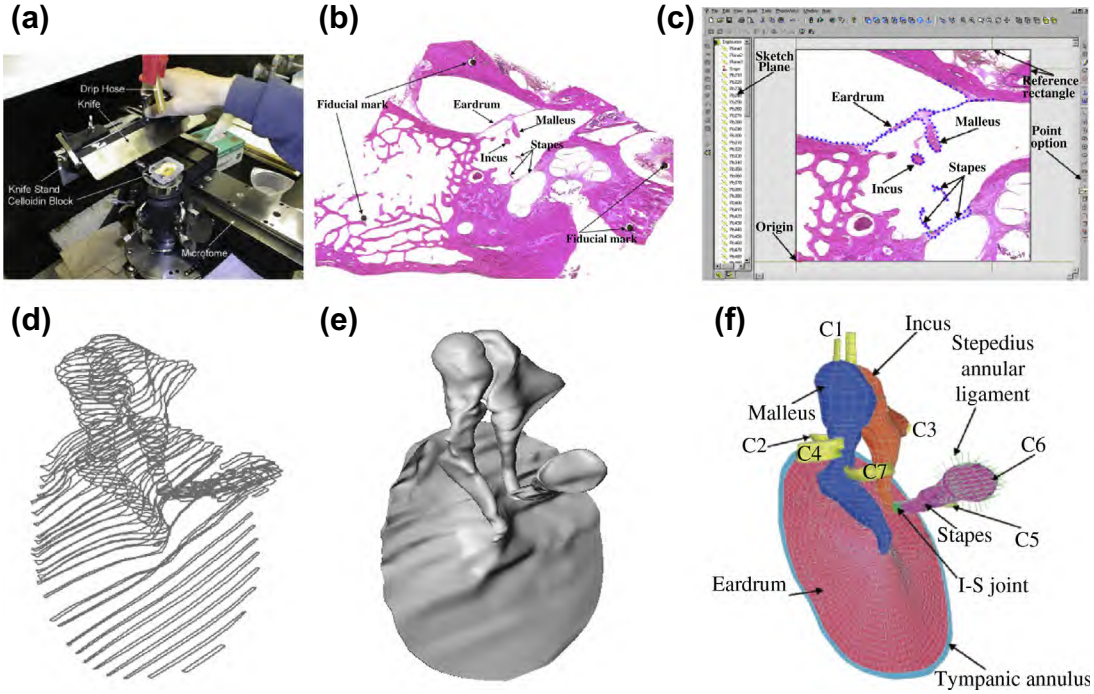


FIGURE 2.36

Construction of the middle ear finite element model: (a) temporal bone slicing and slide preparations; (b) sample section image with fiducial marks for alignment; (c) image digitization; (d) section curves; (e) smooth solid model; (f) finite element model (Sun et al., 2002).

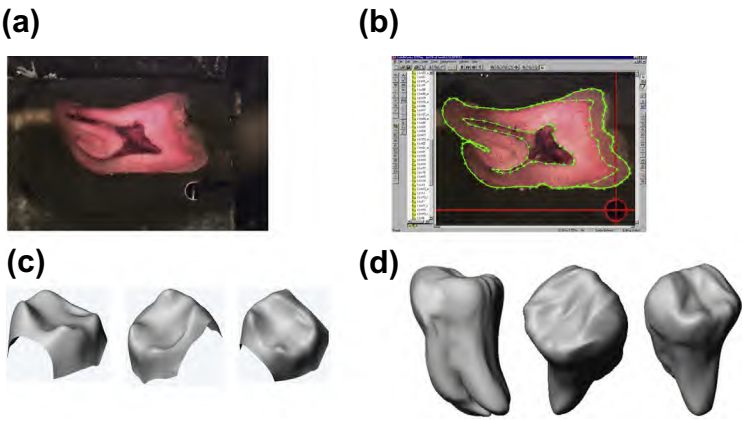


FIGURE 2.37

Geometric model construction for a human tooth. (a) A sample section image. (b) Section sketch digitization with references. (c) Surface model of the DEJ. (d) The solid model in various views.

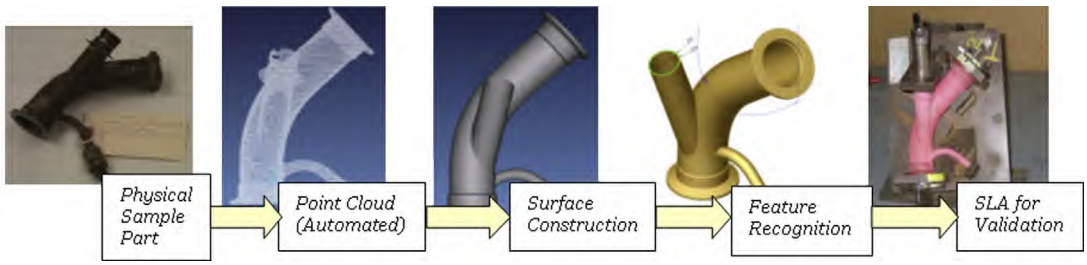


FIGURE 2.38

Reverse engineering of airplane engine tubing.

2.7 SUMMARY

In this chapter, we discussed basic and essential topics in geometric modeling, including parametric representations for curves and surfaces. We discussed popular curve and surface formats, including the most versatile and general NURB curves and surfaces, which are widely employed for geometric modeling. We hope the discussion became directly relevant when we introduced the surfaces generated by CAD. We also include the topic of geometric transformation in this chapter, which is essential to understand how the geometric entities are transformed to support numerous needs in modeling. Detailed derivations were provided in this chapter because geometric modeling serves as the foundation for solid modeling in CAD, which is at the center of the e-Design paradigm. In addition to the mathematical forms of the curves and surfaces, we include curve fitting and surface skinning techniques, which are powerful for many engineering applications. We hope by now you have a fine understanding of the basics of geometric modeling, as we move to the next chapter to discuss solid modeling and CAD theory. With a good understanding of solid modeling, we will then discuss CAD assembly in Chapter 4 and then move into the heart of Part 1 Product Design Modeling—design parameterization for part and assembly in Chapter 5.

APPENDIX 2A: BASIS FUNCTIONS OF B-SPLINE CURVES AND SURFACES

In this appendix, we provide detailed derivations that lead to the six basis functions $N_{i,3}(u)$, $i = 0, 5$, of a quadratic B-spline curve, as stated in Eqs 2.48a–f.

Recall that the basis functions $N_{i,k}(u)$ are defined recursively as

$$N_{i,k}(u) = \frac{(u - t_i)N_{i,k-1}(u)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - u)N_{i+1,k-1}(u)}{t_{i+k} - t_{i+1}} \tag{2.44}$$

and

$$N_{i,1}(u) = \begin{cases} 1, t_i \le u \le t_{i+1} \\ 0, \text{ elsewhere.} \end{cases} \tag{2.45}$$

Note that t is called knots in Eqs 2.44 and 2.45, defined as

$$\mathrm{t}_i = \begin{cases} 0, & i < k \\ i - k + 1, & k \le i \le n \\ n - k + 2, & i > n \end{cases} \tag{2.46}$$

There are $n + k + 1 = 5 + 2 + 1 = 9$ knots. Also, the knots of the curve are

$$\begin{aligned} \mathrm{t}_{0,1,2} &= 0 \\ \mathrm{t}_3 &= 1 \\ \mathrm{t}_4 &= 2 \\ \mathrm{t}_5 &= 3 \\ \mathrm{t}_{6,7,8} &= 4. \end{aligned} \tag{2.47}$$

From Eq. 2.45, we have

$$\begin{aligned} \mathrm{N}_{0,1}(u) &= \mathrm{N}_{1,1}(u) = 0 \\ \mathrm{N}_{2,1}(u) &= 1, \quad \text{for } 0 \le u \le 1 \\ \mathrm{N}_{3,1}(u) &= 1, \quad \text{for } 1 \le u \le 2 \\ \mathrm{N}_{4,1}(u) &= 1, \quad \text{for } 2 \le u \le 3 \\ \mathrm{N}_{5,1}(u) &= 1, \quad \text{for } 3 \le u \le 4 \\ \mathrm{N}_{6,1}(u) &= \mathrm{N}_{7,1}(u) = 0 \end{aligned} \tag{2A.1}$$

which are step functions, also called switch functions.

Then, for $k = 2$, from Eq. 2.44 we have

$$\mathrm{N}_{i,2}(u) = \frac{(u - \mathrm{t}_i)\mathrm{N}_{i,1}(u)}{\mathrm{t}_{i+1} - \mathrm{t}_i} + \frac{(\mathrm{t}_{i+2} - u)\mathrm{N}_{i+1,1}(u)}{\mathrm{t}_{i+2} - \mathrm{t}_{i+1}} \tag{2A.2}$$

and

$$\begin{aligned} \mathrm{N}_{0,2}(u) &= \frac{(u - \mathrm{t}_0)\mathrm{N}_{0,1}(u)}{\mathrm{t}_1 - \mathrm{t}_0} + \frac{(\mathrm{t}_2 - u)\mathrm{N}_{1,1}(u)}{\mathrm{t}_2 - \mathrm{t}_1} = 0 + 0 = 0 \\ \mathrm{N}_{1,2}(u) &= \frac{(u - \mathrm{t}_1)\mathrm{N}_{1,1}(u)}{\mathrm{t}_2 - \mathrm{t}_1} + \frac{(\mathrm{t}_3 - u)\mathrm{N}_{2,1}(u)}{\mathrm{t}_3 - \mathrm{t}_2} = 0 + (\mathrm{t}_3 - u)\mathrm{N}_{2,1}(u) = (1 - u)\mathrm{N}_{2,1}(u) \\ \mathrm{N}_{2,2}(u) &= \frac{(u - \mathrm{t}_2)\mathrm{N}_{2,1}(u)}{\mathrm{t}_3 - \mathrm{t}_2} + \frac{(\mathrm{t}_4 - u)\mathrm{N}_{3,1}(u)}{\mathrm{t}_4 - \mathrm{t}_3} = \mathrm{uN}_{2,1}(u) + (2 - u)\mathrm{N}_{3,1}(u) \\ \mathrm{N}_{3,2}(u) &= \frac{(u - \mathrm{t}_3)\mathrm{N}_{3,1}(u)}{\mathrm{t}_4 - \mathrm{t}_3} + \frac{(\mathrm{t}_5 - u)\mathrm{N}_{4,1}(u)}{\mathrm{t}_5 - \mathrm{t}_4} = (u - 1)\mathrm{N}_{3,1}(u) + (3 - u)\mathrm{N}_{4,1}(u) \\ \mathrm{N}_{4,2}(u) &= \frac{(u - \mathrm{t}_4)\mathrm{N}_{4,1}(u)}{\mathrm{t}_5 - \mathrm{t}_4} + \frac{(\mathrm{t}_6 - u)\mathrm{N}_{5,1}(u)}{\mathrm{t}_6 - \mathrm{t}_5} = (u - 2)\mathrm{N}_{4,1}(u) + (4 - u)\mathrm{N}_{5,1}(u) \\ \mathrm{N}_{5,2}(u) &= \frac{(u - \mathrm{t}_5)\mathrm{N}_{5,1}(u)}{\mathrm{t}_6 - \mathrm{t}_5} + \frac{(\mathrm{t}_7 - u)\mathrm{N}_{6,1}(u)}{\mathrm{t}_7 - \mathrm{t}_6} = (u - 3)\mathrm{N}_{5,1}(u) + 0 = (u - 3)\mathrm{N}_{5,1}(u) \\ \mathrm{N}_{6,2}(u) &= 0. \end{aligned} \tag{2A.3}$$

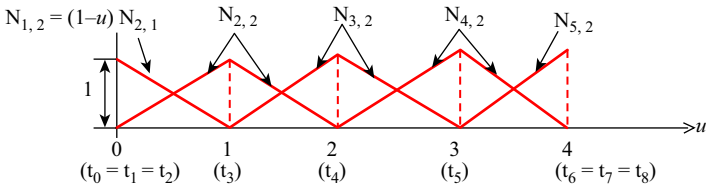


FIGURE 2A.1

Basis functions $N_{i,2}(u)$.

These are piecewise linear functions, as shown in Figure 2A.1.
Now, for $k = 3$, from Eq. 2.48, we have

$$N_{i,3}(u) = \frac{(u - t_i)N_{i,2}(u)}{t_{i+2} - t_i} + \frac{(t_{i+3} - u)N_{i+1,2}(u)}{t_{i+3} - t_{i+1}} \tag{2A.4}$$

and

$$\begin{aligned}
N_{i,3}(u) &= \frac{(u - t_i)N_{i,2}(u)}{t_{i+2} - t_i} + \frac{(t_{i+3} - u)N_{i+1,2}(u)}{t_{i+3} - t_{i+1}} \\
N_{0,3}(u) &= \frac{(u - t_0)N_{0,2}(u)}{t_2 - t_0} + \frac{(t_3 - u)N_{1,2}(u)}{t_3 - t_1} = (1 - u)^2 N_{2,1}(u) \\
N_{1,3}(u) &= \frac{(u - t_1)N_{1,2}(u)}{t_3 - t_1} + \frac{(t_4 - u)N_{2,2}(u)}{t_4 - t_2} = \frac{1}{2}u(4 - 3u)N_{2,1}(u) + \frac{1}{2}(2 - u)^2 N_{3,1}(u) \\
N_{3,3}(u) &= \frac{(u - t_3)N_{3,2}(u)}{t_5 - t_3} + \frac{(t_6 - u)N_{4,2}(u)}{t_6 - t_4} \\
&= \frac{1}{2}(u - 1)^2 N_{3,1}(u) + \frac{1}{2}\left(-2u^2 + 10u - 11\right)N_{4,1}(u) + \frac{1}{2}(4 - u)^2 N_{5,1}(u) \\
N_{4,3}(u) &= \frac{(u - t_4)N_{4,2}(u)}{t_6 - t_4} + \frac{(t_7 - u)N_{5,2}(u)}{t_7 - t_5} \\
&= \frac{1}{2}(u - 2)^2 N_{4,1}(u) + \frac{1}{2}\left(-3u^2 + 20u - 32\right)N_{5,1}(u) \\
N_{5,3}(u) &= \frac{(u - t_5)N_{5,2}(u)}{t_7 - t_5} + \frac{(t_8 - u)N_{6,2}(u)}{t_8 - t_6} = \frac{1}{2}(u - 3)^2 N_{5,1}(u).
\end{aligned} \tag{2A.5}$$

These are the quadratic functions shown in Figure 2.12.

APPENDIX 2B: REPRESENTING CONICS WITH QUADRATIC NURB CURVES

In Example 2.10, we showed that a quadratic NURB curve with three control points represents a 90° circular arc analytically, in which we set the weights $h_0 = h_2 = 1$, and $h_1 = \frac{1}{\sqrt{2}}$. In this appendix we explain why such weights turns a NURB curve into a circular arc. We provide the explanation in a

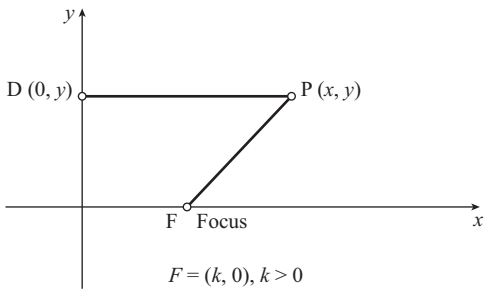


FIGURE 2B.1

Basics of constructing a conic curve.

broader sense, extending the topic to include the entire conics family. We hope that by doing so we offer a more comprehensive explanation on this important topic.

First, in analytic geometry, a conic may be defined as a planar algebraic curve of degree 2, which is written as an implicit equation of degree 2 as follows:

$$f(x,y) = \mathrm{A}x^2 + 2\mathrm{B}xy + \mathrm{C}y^2 + 2\mathrm{D}x + 2\mathrm{E}y + 1 = 0 \tag{2B.1}$$

Geometrically, a conic is the locus of a point moving on the x–y plane so that its distance from a fixed point (called the focus, point F, in Figure 2B.1) is proportional to its distance to a fixed line (called the directrix, usually the y-axis). As shown in Figure 2B.1, the focus F is located at $(k,0)$; any point on the directrix, such as point D, can be represented as $(0,y)$. The locus of the conics must satisfy the proportionality e, called eccentricity, defined as

$$e = \frac{\mathrm{FP}}{\mathrm{PD}}. \tag{2B.2}$$

From Figure 2B.1, the eccentricity e can be written as follows:

$$e = \frac{\sqrt{(x-k)^2 + y^2}}{|x|} \tag{2B.3}$$

Square both sides and arrange terms, which yields

$$(1-e^2)x^2 - 2kx + y^2 + k^2 = 0. \tag{2B.4}$$

When $e = 1$, Eq. 2B.4 becomes

$$-2kx + y^2 + k^2 = 0 \tag{2B.5}$$

which is a parabola (see example in Figure 2B.2(a)). When $e < 1$, the coefficient of the x^2 term in Eq. 2B.4 is positive, and the equation becomes

$$(1-e^2)x^2 - 2kx + y^2 + k^2 = 0 \tag{2B.6}$$

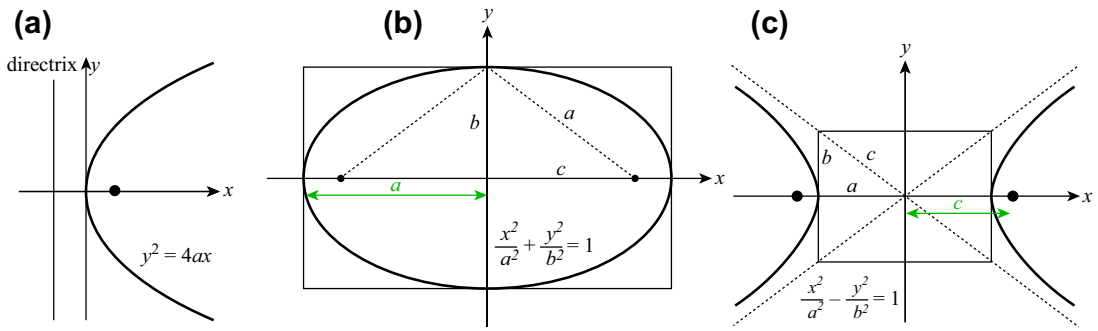


FIGURE 2B.2

Conic curves. (a) A parabola. (b) An ellipse. (c) A hyperbola.

which can be converted into a form

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{2B.7}$$

which represents an ellipse (see example in Figure 2B.2(b)). Eq. 2B.7 represents a circle when $a = b$. When $e > 1$, the coefficient of the x^2 term in Eq. 2B.4 is negative, and the equation becomes

$$-(e^2 - 1)x^2 - 2kx + y^2 + k^2 = 0 \tag{2B.8}$$

which can be converted into a form

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \tag{2B.9}$$

which represents a hyperbola (see example in Figure 2B.2(c)).

With a basic understanding of conics, we proceed with representing a conic curve with a NURB curve.

First, a NURB of degree 2 defined by three noncollinear control points $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$, can be a segment of a parabola, as depicted in Figure 2B.3(a). We wish to extend this concept to define ellipse and hyperbola segments. It is well known that a conic curve that passes through $\mathbf{P}_0$ and $\mathbf{P}_2$ and is tangent to $\mathbf{P}_0\mathbf{P}_1$ and $\mathbf{P}_1\mathbf{P}_2$ at $\mathbf{P}_0$ and $\mathbf{P}_2$, respectively (see Figure 2B.3(a)), can be represented by an implicit equation of degree 2 as shown in Eq. 2B.1. Note that there are five unknown in Eq. 2B.1: A, B, C, D, and E. These five unknowns must be determined by five linearly independent equations. Four of these equations can be found by plugging $\mathbf{P}_0$ and $\mathbf{P}_2$ into Eq. 2B.1 and then taking the derivative of Eq. 2B.1 and applying the condition of curve tangency to line segments $\mathbf{P}_0\mathbf{P}_1$ and $\mathbf{P}_1\mathbf{P}_2$.

Each of these four equations is linear in the unknowns A, B, C, D, and E. If we could find one more condition to generate one more needed linear equation, we will have five linear equations with five unknowns. Solving this system of linear equations yields all five coefficients and the conic curve is uniquely determined.

A very natural addition would be one more point. Plugging the coordinates of this point into Eq. 2B.1 will give us an equation that is similar to those for control points $\mathbf{P}_0$ and $\mathbf{P}_2$. This point should be

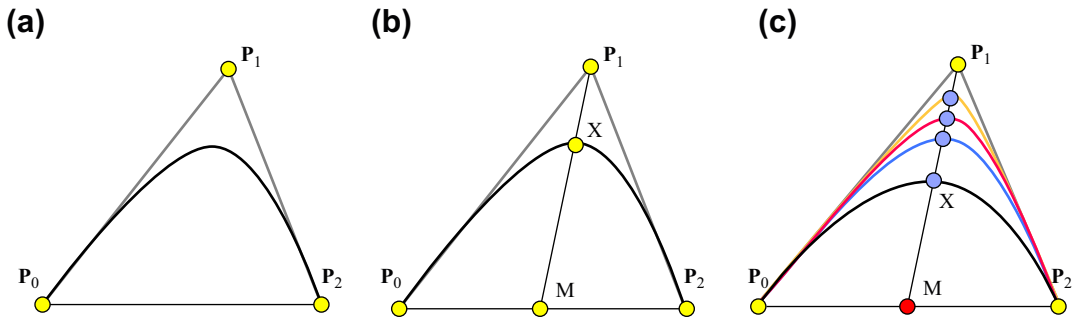


FIGURE 2B.3

Representing conic curves using NURB. (a) A parabola. (b) Point **X** on line segment $\mathbf{P}_1\mathbf{M}$. (c) Different types of conic curves determined by the position of point **X**.

inside the triangle of the three control points so that the convex hull property can be maintained. The position of this point should also be easily changed to produce a different conic curve. One way to do this is by allowing this point to be on the line segment joining $\mathbf{P}_1$ and the midpoint of $\mathbf{P}_0\mathbf{P}_2$ (point M shown in Figure 2B.3(b)). In this way, moving the point **X** on this line segment generates different conic curves, as shown in Figure 2B.3(c).

Recall that the equation of a quadratic NURB curve is

$$\mathbf{P}(u) = \frac{\sum_{i=0}^{2} h_i \mathbf{P}_i N_{i,3}(u)}{\sum_{i=0}^{2} h_i N_{i,3}(u)} = \frac{h_0\mathbf{P}_0 N_{0,3}(u) + h_1\mathbf{P}_1 N_{1,3}(u) + h_2\mathbf{P}_2 N_{2,3}(u)}{h_0 N_{0,3}(u) + h_1 N_{1,3}(u) + h_2 N_{2,3}(u)}. \tag{2B.10}$$

Note that moving point **X** has the same effect as changing the weight h_1 associated with the control point $\mathbf{P}_1$ of the NURB curve. Also, we assumed $h_0 = h_2 = 1$ to ensure the curve tangency at $\mathbf{P}_0$ and $\mathbf{P}_2$, respectively.

If we put $\mathbf{P}_0$ and $\mathbf{P}_2$ on the opposite sides of the x-axis, with the midpoint of $\mathbf{P}_0\mathbf{P}_2$ being the coordinate origin (by a simple translation), we have $\mathbf{P}_0 = -\mathbf{P}_2$. Let the NURB curve meet the line segment $\mathbf{MP}_1$ at **X** as shown in Figure 2B.3(b). A simple calculation using the quadratic NURB curve equation shown in Eq. 2B.10 yields the following:

$$\mathbf{P}(0.5) = \frac{h_1}{1+h_1}\mathbf{P}_1 \quad \text{or} \quad \frac{\mathbf{P}(0.5)}{\mathbf{P}_1} = \frac{h_1}{1+h_1} \tag{2B.11}$$

In other words, from Eq. 2B.3b, we have the following important relationship:

$$\frac{\mathbf{MX}}{\mathbf{MP}_1} = \frac{h_1}{1+h_1} \tag{2B.12}$$

Now, for the quarter circle shown in Figure 2.16(b), or any circular arc like that of Figure 2B.4, we have

$$\mathbf{MX} = \mathbf{OX} - \mathbf{OM} = r - r\sin a = r(1 - \sin a)$$

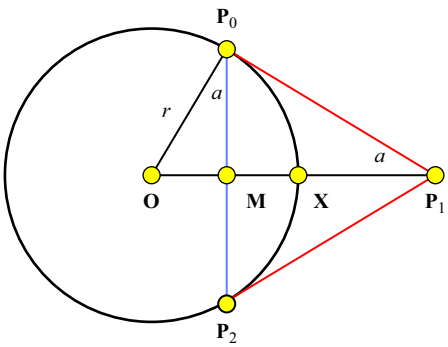


FIGURE 2B.4

A circular arc being defined by a quadratic NURB with three control points.

and

$$\mathbf{MP}_1 = \mathbf{OP}_1 - \mathbf{OM} = \frac{r}{\sin a} - r\sin a = \frac{r\left(1-\sin^2 a\right)}{\sin a}.$$

From the above two equations, we have

$$\frac{\mathrm{h}_1}{1+\mathrm{h}_1} = \frac{\mathbf{MX}}{\mathbf{MP}_1} = \frac{\sin a}{1+\sin a}$$

which implies $\mathrm{h}_1 = \sin a$ as desired.

QUESTIONS AND EXERCISES

2.1. Verify that the functions in Eq. 2.3 are indeed representing a circle. Plot points $(x(u), y(u))$ using a program, such as Matlab, or write a computer program to do so.

2.2. Given two points, $\mathbf{P}_0$, $\mathbf{P}_1$, and a tangent vector at the start point $\mathbf{P}_{0,u}$, derive equations for a parametric quadratic curve that passes through these two points at $u = 0$ and 1, respectively, with tangent vector $\mathbf{P}_{0,u}$ at $u = 0$.

Graph the basis functions in Matlab, make observations on the characteristics of the function, and comment on their influence on the curve geometry.

Graph the curve for

$\mathbf{P}_0 = [0,1]$, $\mathbf{P}_1 = [3,2]$,$\mathbf{P}_{0,u} = [2,-7]$.

In addition to the derivations, submit screen captures of Matlab graphs to show the curve and basis functions.

2.3. Given three points, $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, and the tangent vector at the end point $\mathbf{P}_{2,u}$, derive equations for a parametric cubic curve that passes through these three points at $u = 0$, ½, and 1, respectively, with tangent vector $\mathbf{P}_{2,u}$ at $u = 1$.

Graph the basis functions and comment on their influence on the curve geometry.

Graph the curve for:

$\mathbf{P}_0 = [0,1]$, $\mathbf{P}_1 = [2,0]$,$\mathbf{P}_2 = [3,2]$,$\mathbf{P}_{2,u} = [2,-7]$.

Submit screen captures of the Matlab graph to show the curve and basis functions.

2.4. Continue from Problem 3. Calculate the tangent vectors of the curve at both start and end points, following curve format conversion. Calculate the positions of the curve points at $u = 1/3$ and 2/3, following curve format conversion. Calculate the position of the interior control points $\mathbf{P}_1$ and $\mathbf{P}_2$ of the equivalent Bézier curve using curve format conversion.

2.5. Four control points on the x–y plane are given as follows:
$\mathbf{P}_0 = [0,0]$, $\mathbf{P}_1 = [1,4]$, $\mathbf{P}_2 = [2,-5]$, $\mathbf{P}_3 = [3,8]$.

- **a.** Construct a Bézier curve enclosed by the control polygon formed by the four given points;
- **b.** Graph the curve in Matlab and submit screen captures to show the curve and basis functions.

2.6. Show that the 4×4 $\mathbf{M}^4$ matrix of a cubic uniform B-spline curve defined as

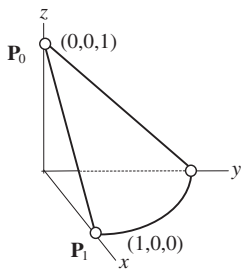
$$\mathbf{P}^i(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}_{1\times4} \mathbf{M}^4_{4\times4} \begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \end{bmatrix} = \mathbf{U}_{1\times4} \mathbf{M}^4_{4\times4} \begin{bmatrix} \mathbf{P}_{i-1} \\ \mathbf{P}_i \\ \mathbf{P}_{i+1} \\ \mathbf{P}_{i+2} \end{bmatrix}, \quad u \in [0,1], i \in [1, n-2]$$

is

$$\mathbf{M}^4_{4\times4} = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}.$$

2.7. Show that the closed uniform B-spline curve of cubic basis functions is C^2-continuous. Calculate the Cartesian coordinates of the start and end points of all curve segments of Example 2.9, both quadratic and curve B-spline curves.

2.8. Derive a parametric equation for the surface of the quarter cone, using the following:

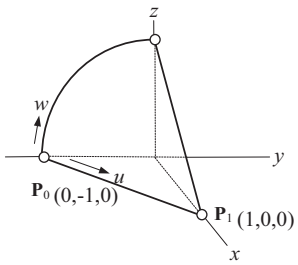


- **a.** Surface of revolution. Plot the surface using Matlab, and
- **b.** Sweep surface. Note that
  $\mathbf{P}_0 = [0,0,1]$, $\mathbf{P}_1 = [1,0,0]$.

Submit the following:

- **c.** Detailed equations that describe the surface of revolution and sweep surface;
- **d.** Matlab scripts and screen captures of the surface plotted in Matlab.

2.9. Derive a parametric equation for the surface of the quarter cone shown below, using the surface of revolution. Plot the surface using Matlab. Note that



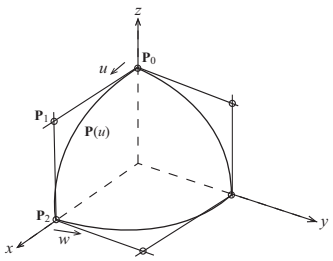
$\mathbf{P}_0 = [0,-1,0]$, $\mathbf{P}_1 = [1,0,0]$.

Submit the following:

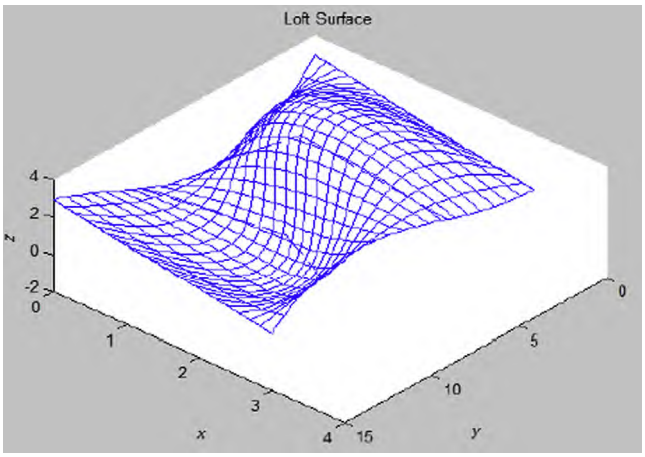
a. Detailed parametric equations that describe the surface;

b. Matlab script and screen capture of the surface plotted in Matlab.

2.10. Derive a parametric equation for a 1/8 sphere of radius 1 shown below formed by revolving a quadratic NURB curve $\mathbf{P}(u)$ on the x–z plane along the z-axis.



2.11. Derive a parametric equation for a blend surface formed by four curves. These four curves are:



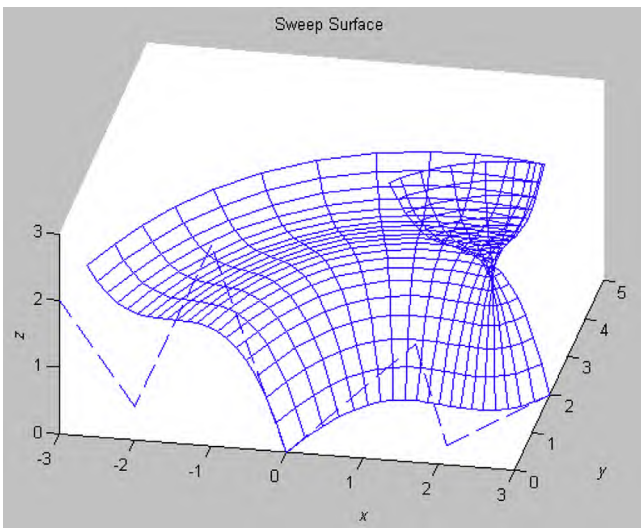
a. Straight with end points:
$\mathbf{G}_0 = [0,0,3]$, $\mathbf{G}_1 = [3,0,1]$;
b. Spline curve with three points:
$\mathbf{P}_0 = [0,5,0]$, $\mathbf{P}_1 = [1,5,3]$, $\mathbf{P}_2 = [3,5,2]$;
c. Bézier curve with four control points:
$\mathbf{Q}_0 = [0,10,1]$, $\mathbf{Q}_1 = [2,10,2]$, $\mathbf{Q}_2 = [2.5,10,0.5]$, $\mathbf{Q}_3 = [3,10,3]$.
d. Straight line with end points:
$\mathbf{R}_0 = [0,15,3]$, $\mathbf{R}_1 = [3,15,1]$.

Also, create a solid (or surface) feature using these four curves in Pro/ENGINEER or SolidWorks. Submit the following:

e. Screen capture of the Pro/ENGINEER or SolidWorks model and sketch view with all four sections;
f. Detailed equations that describe the surface;
g. Matlab script and screen capture of the surface plotted in Matlab.

2.12. Derive a parametric equation for a sweep surface formed by sweeping a cubic Bézier curve $\mathbf{P}(u)$ on the x–z plane along a trajectory of the same curve $\mathbf{Q}(u)$ on the x–y plane. The control points of these two curves are, respectively:
$\mathbf{P}_0 = [0,0,0]$, $\mathbf{P}_1 = [-1,0,3]$, $\mathbf{P}_2 = [-2,0,0.5]$, $\mathbf{P}_3 = [-3,0,2]$;
$\mathbf{Q}_0 = [0,0,0]$, $\mathbf{Q}_1 = [1,3,0]$, $\mathbf{Q}_2 = [2,0.5,0]$, $\mathbf{Q}_3 = [3,2,0]$.
Also, create a sweep solid feature using these two curves in Pro/ENGINEER or SolidWorks. Submit the following:

a. Screen capture of the Pro/ENGINEER or SolidWorks solid model and a sketch view of both curves;
b. Detailed equations that describe the surface;
c. Matlab script and screen capture of the surface plotted in Matlab.



REFERENCES

Bendsoe, M.P., Sigmund, O., 2003. Topology Optimization, Theory, Methods, and Applications. Springer-Verlag, Berlin, Heidelberg, New York. ISBN 3-540-42992-1.

Chang, K.H., Siddique, Z., Edke, M., Chen, Z., 2006. An integrated testbed for reverse engineering of aging systems and components. Computer-aided Design and Applications 3 (1–4), 21–30.

Chang, K.H., Magdum, S., Khera, S., Goel, V.K., May 2003. An advanced computer modeling and prototyping method for human tooth mechanics study. Annals of Biomedical Engineering 31 (5), 621–631.

Chang, K.H., Choi, K.K., 1992. A geometry-based parameterization method for shape design of elastic solids. Mechanics of Structures and Machines 20 (2), 215–252.

Mortenson, M.E., 2006. Geometric Modeling, third ed. Industrial Press.

Piegl, L., Tiller, W., 1987. Curve and surface construction using rational B-splines. Computer-aided Design 19 (9), 485–498.

Sun, Q., Chang, K.H., Dormer, K., Dyer, R., Gan, R.Z., November 2002. An advanced computer-Aided geometric modeling and fabrication method for human middle ear. Medical Engineering and Physics 24 (9), 595–606.

Tang, P.-S., Chang, K.H., 2001. Integration of topology and shape optimizations for design of structural components. Journal of Structural Optimization 22 (1), 65–82.

CHAPTER

SOLID MODELING

3

CHAPTER OUTLINE

3.1 Introduction 126
3.2 Basics of Solid Modeling 127
3.2.1 Wireframe Models 127
3.2.2 Surface Models 129
3.2.3 Solid Models 130
3.2.4 Major Modeling Schemes 132
3.2.4.1 Constructive Solid Geometry 132
3.2.4.2 Boundary Representation 135
3.3 Feature-Based Parametric Solid Modeling 139
3.3.1 Geometric Features 140
3.3.2 Sketch Profiles 142
3.3.2.1 Sketch Relations 142
3.3.2.2 Variational Modeling 144
3.3.3 Parent–Child Relationships 149
3.3.4 Parametric Modeling 150
3.3.5 Solid Modeling Procedure in CAD 151
3.3.6 Direct Modeling 155
3.3.7 Geometric Modeling Kernels 155
3.4 Solid Model Build Plan 157
3.5 Commercial CAD Systems 160
3.5.1 General Purpose Codes 161
3.5.2 Special Codes 161
3.6 Summary 162
Appendix 3A: Sketch Relations 162
Questions and Exercises 164
References 167

With the basic understanding of geometric modeling discussed in Chapter 2, we are moving closer to the core of product design modeling—that is, solid modeling, especially feature-based parametric solid modeling, which is the key topic to be discussed in this chapter. In recent decades, the term *solid modeling* has been associated with the technology of using computer-aided design (CAD) systems to create the shape and form of part geometry and associated physical properties with a computer for the purpose of engineering designs. Today, CAD models with built-in essential product design information play a central role in e-Design.

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00003-X
Copyright © 2015 Elsevier Inc. All rights reserved.

Solid modeling technology in CAD applications has evolved through a series of phases in order to improve the geometric representation of physical artifacts or design concepts that are being developed in the engineering design process. It started in the early 1960s when the first wireframe computer graphic was invented at the Massachusetts Institute of Technology (MIT) Lincoln Laboratory. In the mean time, design automated by computer (DAC-1), the first production interactive graphics manufacturing system, was developed by General Motors. Since then, with further development, surface modeling became a reality in the 1960s and solid modeling began in the 1970s, followed by feature modeling in the mid-1980s and parametric modeling by Parametric Technology in the late 1980s. With the development of feature modeling and parametric modeling in the 1980s, feature-based parametric solid modeling has since become the mainstream CAD theory in support of engineering design. It is well recognized that the most significant development appeared in the mid-1990s, in which major CAD tools were made available in personal computers (PCs) that allowed end users in mid- and small-size companies to be able to bring designs from the drawing board into digital form. More recently, direct modeling technology brought CAD one step further in support of engineering design by allowing designers to directly manipulate solid models by pulling or squeezing solid features on the computer screen using a mouse.

Today, major CAD systems employ feature-based parametric modeling techniques to support engineering designs through respective interactive user interfaces. Because solid modeling is the heart and soul of CAD, we devote this chapter to introducing basic knowledge in solid modeling methods and theory. This is the knowledge that readers must have in order to proceed with the study of the e-Design paradigm and gain practical skills in practicing e-Design, in which product geometry is represented in CAD solid models throughout the product development process.

This chapter is organized with the assumption that the reader has used CAD software (e.g., SolidWorks, Pro/ENGINEER) for creating solid models but has no or little background in solid modeling theory. If you are not familiar with CAD software, you are strongly encouraged to review excellent references for tutorial lessons, such as Toogood and Zecher (2012) for Pro/ENGINEER or Planchard and Planchard (2013) for SolidWorks. With the assumption that you are familiar with CAD software, we offer discussion on numerous topics involved in solid modeling, with examples extracted mostly from SolidWorks and Pro/ENGINEER.

Overall, the objectives of this chapter are (1) to provide an introduction to the basic solid modeling theories that help readers understand how the product design is realized in CAD, and (2) to help readers become familiar with the behind-the-scenes operations in CAD modeling so as to effectively use these tools for design. We also provide a short discussion on commercial CAD software tools, with the hope of offering readers guidance on selecting proper tools that are suitable for their specific needs.

3.1 INTRODUCTION

In the 1970s, nearly every engineering drawing produced in the world was done with pencil or ink on paper. A drafter leaned on the drawing board and used a T-square ruler, protractors, a compass, and templates to carefully sketch the lines, arcs, letters, and symbols that constitute an engineering drawing. Any changes or mistakes required erasing and redrawing, whereas major changes often necessitated recreation of the drawing from scratch. In manually created drawings, one of the most challenging tasks is that a drafter must envision the intersection of solid entities, unwrap the intersecting curves, and sketch the curves accurately on the drawing paper. Engineering drawing has been the backbone of product design and development for many years (Bozdoc, 2003).

Although engineering drawing still plays an important role in product design and manufacturing in many industrial sectors, manual sketching for creating drawings has been gradually replaced by CAD (computer-aided design) software using computers. Beginning in the 1980s, CAD software reduced the need for draftsmen significantly, especially in small to mid-sized companies. The software's affordability and ability to run on personal computers in the mid-1990s allowed engineers to do their own drafting and analytic work to some extent.

In fact, instead of just creating drawings, CAD has fundamentally changed the way design is done. As in the manual drafting of technical and engineering drawings, the output of CAD conveys information, such as materials, processes, dimensions, and tolerances, according to application-specific conventions in solid models. Instead of drafting in digital form, designers use CAD to create product models in solid model forms with adequate product data, then they create drafting if necessary. CAD solid models offer flexibility and efficiency when making design changes; provide geometric and physical data that support product performance evaluations using computer-aided engineering (CAE); support virtual manufacturing, prototyping, manufacturing process planning, and product cost estimating; and offer product life cycle and product knowledge repository for archiving. Most important, product model in CAD serves as the centerpiece for e-Design.

The backbone of CAD is solid modeling. It is indispensable for designers to acquire adequate knowledge in solid modeling in order to effectively practice e-Design in support of engineering design using CAD software. We introduce numerous theories and schemes that support product (or more specifically, parts) representation in solid models, with a focus on feature-based parametric solid modeling, which is the mainstream solid modeling method offered in major CAD systems. The main theme of the chapter is understanding the behind-the-scenes operations while you are using CAD for creating solid models. It is also important for readers to understand how CAD rebuilds solid models when a design change is made by changing dimension values associated with solid features.

We start in Section 3.2 by introducing the basic theories of solid modeling, including constructive solid geometry (CSG) and boundary representation (B-rep), which are the two most widely used schemes for solid modeling. With a basic understanding of solid modeling, we discuss the main topic of the chapter in Section 3.3—that is, the feature-based parametric solid modeling method. In Section 3.4, we offer the practical aspects of creating solid models by discussing model construction plans. We then provide a short overview of commercial CAD software in Section 3.5.

3.2 BASICS OF SOLID MODELING

Before getting into the main topic of this chapter—feature-based parametric solid modeling—we discuss a few important basic topics in solid modeling in this section. We start by discussing three basic methods for representing solid models: wireframe, surface, and solid forms. We include the advantages and disadvantages of each form, as well as the use of the models represented in the form for design and manufacturing applications. We will then narrow our focus to solid modeling, for which we introduce two major modeling methods: CSG and B-rep.

3.2.1 WIREFRAME MODELS

Wireframe is the simplest and the earliest form of representing physical objects; it was first introduced in 1963 at MIT's Lincoln Laboratory. The wireframe form represents a shape by its characteristic

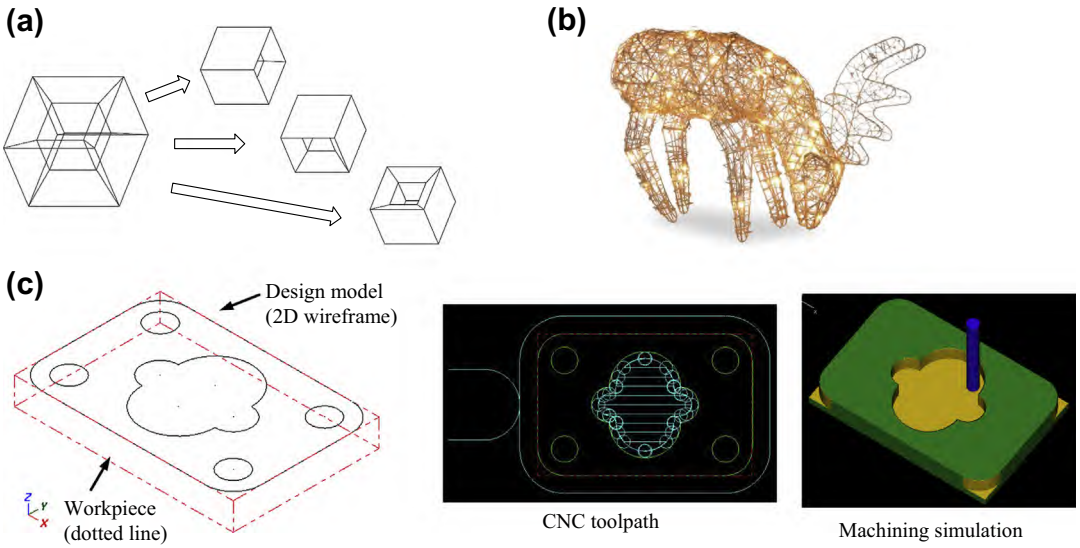


FIGURE 3.1

Wireframe models. (a) Ambiguity in representing a solid object. (b) Outdoor Christmas decoration of a Rattan Reindeer (courtesy of http://www.brookstone.com/pre-lit-outdoor-christmas-decorations-rattan-reindeer). (c) Supporting toolpath generation in Mastercam.

curves (lines, arcs, splines, and so on) and points, as illustrated in Figure 3.1(a). A reindeer frame decoration, as shown in Figure 3.1(b), which is displayed in residential front yards during Christmas, is a good example of a wireframe model in real-world applications. The major advantages of this method are that it requires simple input from users and the modeling software is relatively easy to implement. One of the examples in its applications in design and manufacturing is the two-dimensional (2D) wireframe models in Mastercam that support numerical control (NC) toolpath generation for machining simple prismatic features, such as pocket milling or profile milling (Figure 3.1(c)), in which all contours exist in flat planes and only planar geometric information is required.

Although there are some applications that a wireframe model is able to support, there are major issues involved in representing solid models in wireframe. First, it is ambiguous for representing a solid object in a wireframe, as illustrated in Figure 3.1(a), due to its inability to determine the inside or outside of a solid object. Second, a wireframe is not able to represent objects with nonpolygonal boundaries due to a lack of curvature information on surfaces. In addition, it is impossible to calculate mass properties of a solid object represented in a wireframe form. A wireframe model is not capable of supporting a finite element mesh for structural analysis of a physical object other than beam or truss structures. Generating a toolpath on a nonpolygon surface of a solid model represented in a wireframe is impossible due to its lack of surface geometric information.

Because of these reasons, no CAD tool uses wireframe alone to represent part geometry. Wireframes are only used in CAD as one of the options for visualizing solid objects—that is, the wireframe mode—due to its quick response in displaying objects on the computer screen without rendering.

3.2.2 SURFACE MODELS

A surface can be thought of as an infinitely thin shell stretched over a wireframe. In addition to lines and points, surface models represent a shape by its surface geometry, as illustrated in Figure 3.2. A surface model includes information about the faces and edges of a part. Modeling methods for a surface offered in CAD include protrusions, such as extrude, sweep, loft, and revolve; interpolating points; and knitting and trim, as shown in Figure 3.2. A hot air balloon shown in Figure 3.2(f), is a good example of a surface model in real-world applications. To represent a solid object, the surfaces must form an airtight cavity that replicates the geometry of the object without any gap between them or any dangling surface or line. Surfaces are commonly used to model complex, freeform (or organic) shapes that are commonly found in applications in the automotive, aircraft, mold, and consumer goods industries.

A surface model is good for visualizing complex surfaces and supports NC toolpath generation. As illustrated in Figure 3.3(a), a toolpath was generated on a Coons patch in Mastercam. The surface model is also widely used in finite element analysis (FEA) for thin-shell structures, as shown in Figure 3.3(b), in which an FEA was carried out for a surface model that represents the mid-plane of a thin-shell solid object created in CAD. In addition, a stereolithography (STL) model, which is a surface model consisting of triangular facets that form an airtight cavity representing a solid object, is the de-facto standard for three-dimensional (3D) printing (also called rapid prototyping or solid freeform fabrication). An example of the STL model is given in Figure 3.3(c).

As for modeling solid objects, the surface model generally works well. However, the mass or volume information of a solid object represented in a surface model is hard to determine, partly

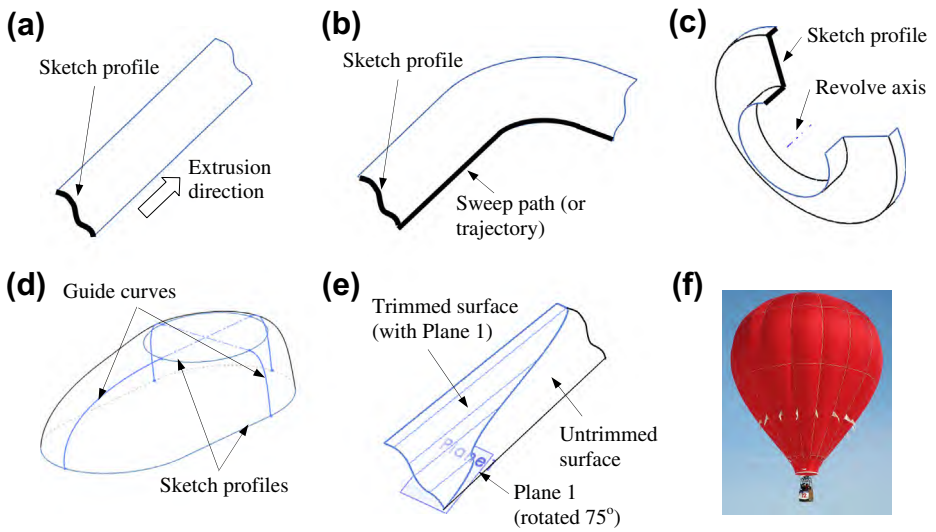


FIGURE 3.2

Creating surface models in computer-aided design. (a) Extrusion. (b) Sweep. (c) Revolve. (d) Loft. (e) Trimmed. (f) A hot air balloon.

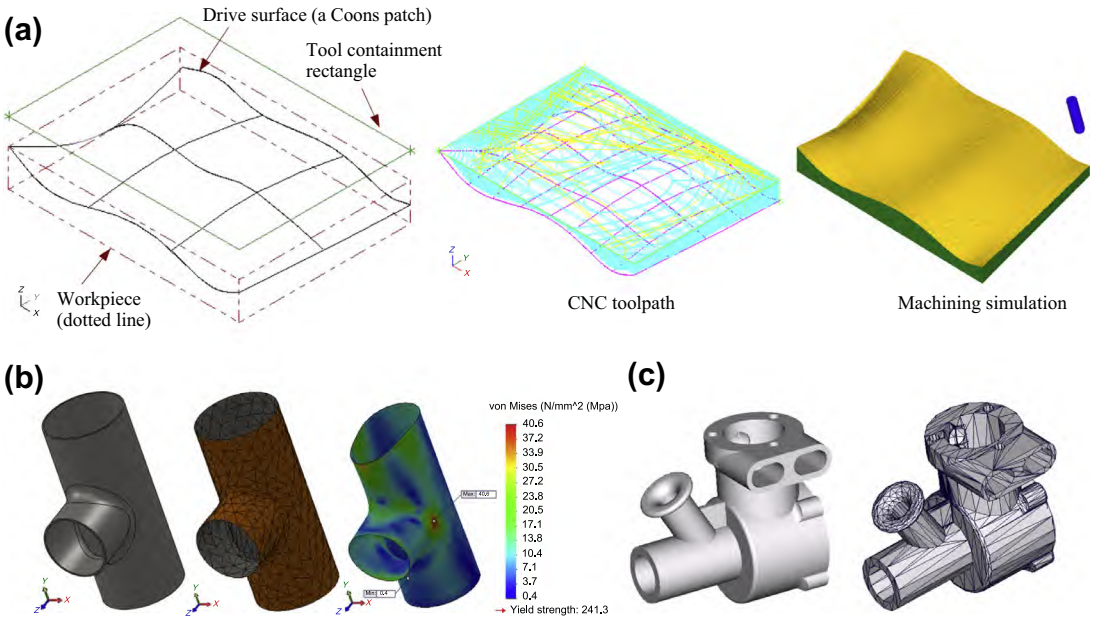


FIGURE 3.3

Surface models for support of design and manufacturing. (a) Toolpath and machining simulation in Mastercam. (b) FEA of a thin-shell structure as a solid model in CAD (left), surface model of mid-plane (middle) with finite element mesh, and FEA stress fringe plot (right). (c) Engine block as a solid model (left) and stereolithography (STL) model (right).

because a surface model lacks the mathematic representation of the solid object. Additional information must be added to a surface model in order to specify in/out and top/bottom of the physical object that the surface model represents.

3.2.3 SOLID MODELS

Solid models contain information about the edges, faces, and the interior of the part. The mathematical description contains information that determines whether any location is inside, outside, or on the boundary surface. Modeling a solid object in the solid model form generally includes primitive creation and Boolean operations, surface operations, protrusion operations, pick-and-place, feature-based modeling, and parametric modeling. It is important to note that individual CAD systems only use some of these methods for the modeling capabilities they respectively offer.

Primitives are basic solid objects with simple mathematical surfaces, as depicted in Figure 3.4(a). These primitives can be controlled by a small number of parameters and positioned using a transformation matrix (as discussed in Chapter 2). Boolean operations, such as union, intersection, and difference, are used to make more complicated objects by combining the basic objects. This method is often referred to as CSG (Constructive Solid Geometry). More about this approach will be discussed in Section 3.2.4.

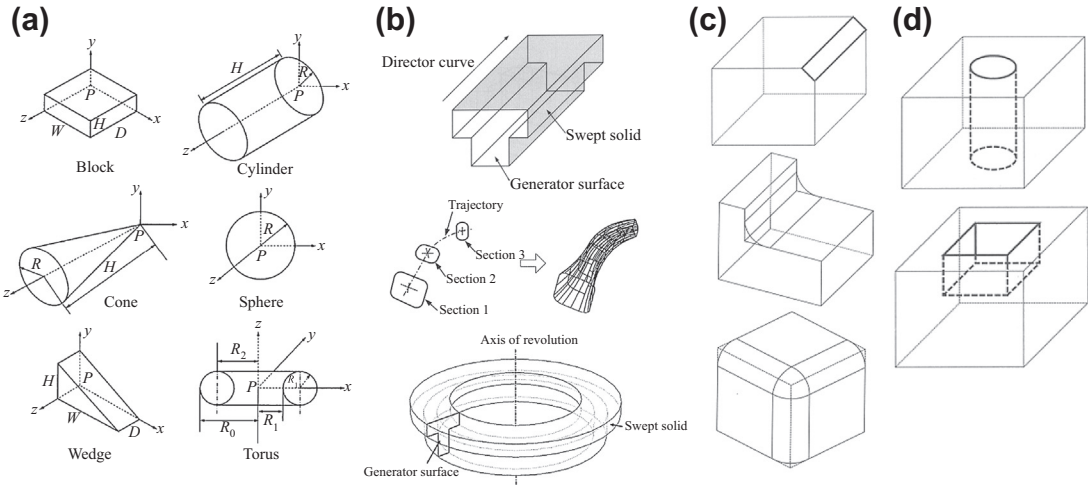


FIGURE 3.4

Solid model construction methods. (a) Primitives and Boolean operations. (b) Protrusion operations. (c) Pick-and-place. (d) Feature operations (Lee, 1999).

Surface operations trim and knit surfaces to form an airtight cavity that represents a solid object. This method is called boundary representation or B-rep. More details of this method are discussed in Section 3.2.4.

Protrusion operations use 2D sketch profiles to generate a 3D solid by extruding, revolving, sweeping, and loft. Examples are shown in Figure 3.4(b).

Pick-and-place operates directly on the solid model surfaces, edges, and vertices to create a desired modification. Some examples include chamfering, rounding/filleting, drafting, and shelling, as illustrated in Figure 3.4(c).

Feature modeling mainly supports manufacturing operations. Manufacturing features are shapes having engineering manufacturing significance. They usually are the geometric embodiment of machining operations, such as hole, pocket, slot, and boss, as illustrated in Figure 3.4(d). Note that *feature* (instead of *manufacturing feature*) is a generic term used by CAD users and developers to refer to almost all kinds of geometric entities in solid modeling, sometimes including nongeometric entities, such as datum features (including planes, axis, points, coordinate systems).

Parametric modeling manipulates parameters to control the geometric shape of a solid object. Parameters come from dimensions in 2D profiles in sketch, dimensions on 3D solid features, and variables in user-defined equations. If defined properly, the entire part geometry can be controlled by a small number of key parameters. Design intents can therefore be captured through the change of the small set of parameters. Parametric modeling supports design parameterization; therefore, it becomes an indispensable part of the product design modeling in the context of e-Design. More about feature-based parametric modeling is discussed in Section 3.3 as a key section of the chapter. Design parameterization, which is an important topic of e-Design, is discussed in Chapter 5.

A solid model is the ultimate way to represent general objects, which are physically solid objects. Solid models support NC toolpath generation of complex surfaces and meshing with solid elements

for finite element analysis. In addition, solid models are adequate for the calculation of mass properties in support of motion simulations. Solid models also support collision and interference checking, which are critical in assembly and kinematic analyses.

3.2.4 MAJOR MODELING SCHEMES

As mentioned above, there are several methods for constructing solid models. From a CAD user's perspective, protrusion and pick-and-place methods are most often employed. In terms of mathematically representing a solid object, two major modeling methods, CSG and B-rep, are widely employed by geometric modeling kernels, which are the core of CAD systems. More about kernels is discussed in Section 3.2.5. In this subsection, we discuss these two modeling methods in greater detail to offer readers a more in-depth understanding of CAD theory and behind-the-scenes mathematic operations.

3.2.4.1 Constructive Solid Geometry

CSG is a modeling method that supports the construction of solid objects through operations on solid primitives. CSG records both the information of operations and information of the primitives. The major components of the CSG method are primitives and instances, as well as the Boolean set operations and CSG tree.

3.2.4.1.1 Primitives and Instances

A typical CSG system uses primitives, such as cylinders, boxes, cones, spheres, as shown in Figure 3.4(a), as well as their instances, as the building blocks for constructing solid models. The idea is to use the primitives that can be manipulated easily in the computer system. Each individual primitive is stored as a geometric family together with a set of parameters. An instance of the primitive family is a scaled and/or transformed replica of its original. The scaling may be uniform or differential. It is a method of keeping primitives in their basic minimum condition, such as unit cube and unit sphere. For example, a cube belongs to the family of box with all three parameters—length, width, and height—having the same value.

3.2.4.1.2 Boolean Set Operations and CSG Tree

To construct a complex object, the CSG approach decomposes it as a compound object composed of a number of primitives. CSG typically uses Boolean set operations, including union, difference, and intersection, to construct objects. These are mathematical operations taken from set theory. The primitives involved in an operation are referred to as operands.

A union of primitives gives a volume occupied by each operand minus the volume shared (or overlapped) by them, as illustrated in Figure 3.5(b), in which union operation is carried out for two primitives P and Q (Figure 3.5(a)). In CAD, such a union operation is realized in many ways. For example, in Figure 3.5(b), you may sketch a profile and extrude it for Q, and extrude (both sides) a square block from Q for P. Implicitly, union operation is employed in CAD.

Difference operations require two operands playing different roles. The base operand defines the source volume (Q in Figure 3.5(c)) and the second operand defines the volume to be removed (P in Figure 3.5(c)). The resulting object contains the volume of the base operand but not the second operand. For example, the object shown in Figure 3.5(c) represents difference operation Q − P. In CAD, you may

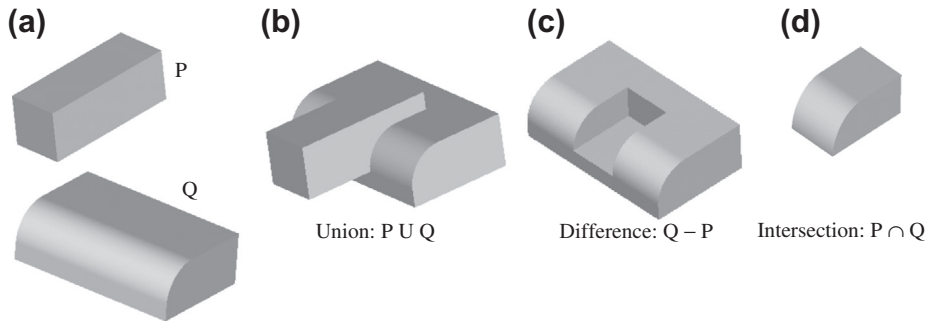


FIGURE 3.5

Boolean operations. (a) Solid primitives P and Q. (b) Union of P and Q. (c) Difference of P and Q. (d) Intersection of P and Q.

create a sketch profile and extrude it for Q, and extrude a square blind cut P to Q to yield the object shown in Figure 3.5(c).

An intersection of primitives gives a volume common to all operands, for example, the object shown in Figure 3.5(d) representing an intersection P ∩ Q. Note that there is no direct intersection operation in major CAD systems.

In fact, as a user, we do not see any direct Boolean operation capabilities offered by CAD systems. However, there are capabilities for users to cut and union features as illustrated in Figure 3.5. In general, location and orientation of primitives involved in the Boolean operations determine largely the resulting object. A primitive or its instance is usually scaled, translated, and rotated to a prescribed location and orientation before carrying out a Boolean operation.

As seen clearly, Boolean operations are binary, involving two (or two sets of) primitives. The primitives involved in each operation and the sequence of operations create a so-called CSG tree. A CSG tree, as shown in Figure 3.6 schematically, is a binary tree with leaf nodes as the primitives and interior nodes (or branch) represent Boolean set operations. The root node represents the final part.

The CSG tree creates a procedural model that specifies how the solid features are combined to form the final solid model. In general, the solid model must be "evaluated" by computing intersecting curves

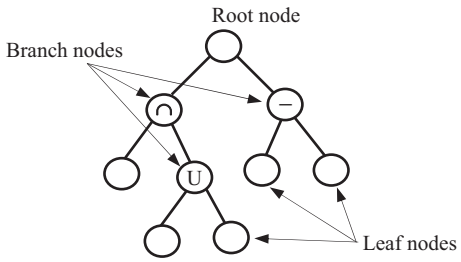


FIGURE 3.6

A schematic of a constructive solid geometry tree.

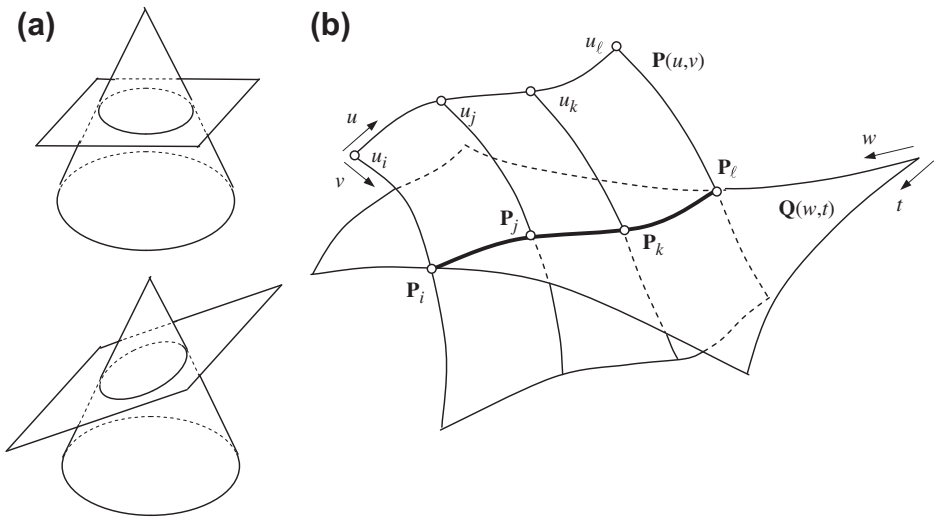


FIGURE 3.7

Surface-to-surface intersection. (a) Intersecting two standard primitives. (b) Intersecting two parametric surfaces.

from the parametric surface equations of the geometric features, based on the position and orientation of the primitives. Quantitative information, such as intersecting curves, must be generated and stored to define and to display the solid model.

The curve intersecting two surfaces can be analytical only for basic primitives, such as a circular cone intersecting with a plane. Conic curves, such as a circle or ellipse, are generated as a result of the intersection, as illustrated in Figure 3.7(a), which can be represented analytically. However, analytical representations of intersecting curves are generally not available, so approximation methods must be used. We discuss one approximation method that calculates intersecting curves of two parametric surfaces as an example.

Consider two parametric surfaces $\mathbf{P}(u,v) = [P_x(u,v), P_y(u,v), P_z(u,v)]$ and $\mathbf{Q}(w,t) = [Q_x(w,t), Q_y(w,t), Q_z(w,t)]$, where u and v, and w and t are parametric coordinates of the two surfaces, respectively (see Figure 3.7(b)). The intersecting curve is constructed by generating a parametric curve that passes through a number of intersection points.

The intersecting curve must satisfy the following equation:

$$\mathbf{P}(u,v) - \mathbf{Q}(w,t) = 0. \tag{3.1}$$

There are four unknowns u, v, w, and t, but only three equations (parametric equations of Eq. 3.1 in the x-, y-, and z-directions, respectively). Basically, the intersecting curve cannot be obtained by solving Eq. 3.1.

One approach to construct the intersecting curve is presented below:

1. Fix a value of, for example, the u parameter of the surface $\mathbf{P}(u,v)$, to generate a curve on the surface (i.e., $\mathbf{P}(u_j,v)$), as shown in Figure 3.7(b).

2. Compute the intersection points of this curve with the surface $\mathbf{Q}(w,t)$ by solving numerically,

$$\mathbf{P}(u_j, v) - \mathbf{Q}(w, t) = 0 \tag{3.2}$$

We have now three equations (parametric equations of Eq. 3.2 in the x-, y-, and z-directions, respectively), and three unknowns (v, w, and t).

3. Repeat Steps 1 and 2 for as many u_j as needed. For example, if a cubic curve is desired, then four intersecting points at $u = 0$, 1/3, 2/3, and 1 can be calculated for a cubic spline curve to approximate the intersecting curve, as illustrated in Figure 3.7(b).

As can be seen from this discussion, CSG is intuitive because the concept of solid model construction is in some sense parallel to manufacturing operations. The solid model constructed is always valid (except if one unions two cones only at their vertices, for example). Another advantage of the CSG is that the solid model requires a small data set (i.e., the primitives involved and CSG tree). In general, the CSG model is an unevaluated model, which must be "evaluated" for numerous purposes (e.g., display on computer screen) and calculating engineering data (e.g., mass properties). As discussed, evaluating a CSG model can be inefficient because the process involves computations, such as calculations of intersecting curves.

3.2.4.2 Boundary Representation

B-rep is an important method of 3D modeling for solid objects. A B-rep model represents a solid object by assembling (or gluing) surfaces to form an "airtight" boundary that encloses the 3D space occupied by the object, as illustrated in Figure 3.8.

In B-rep, a solid model is bounded by faces, a face is bounded by edges, and an edge is bounded by vertices. Essentially, there is a hierarchy of four levels of geometric entities: volume, face, edge, and vertex. Face, edge, and vertex are topology entities that specify connectivity information. In addition to topological entities, the corresponding geometric entities are surface, curve, and point that define the shape, location, and orientation of the entities. Both geometric and topology data must be defined to construct the solid model. Both must be stored in a database.

For example, as shown in Figure 3.9, a block is defined in a volume with its six boundary faces. Each rectangular face is defined by its own edges to form an enclosed surface. All the surfaces are

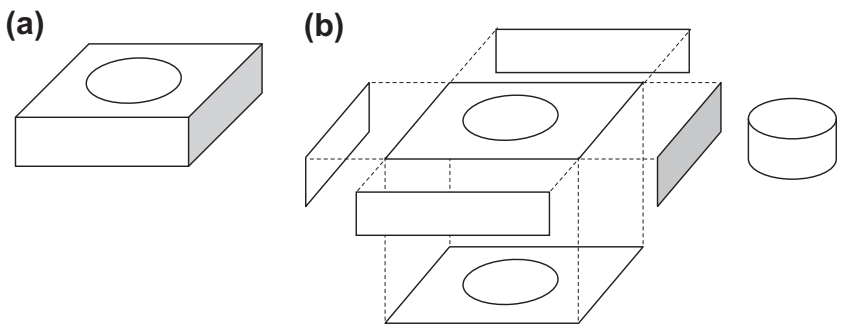


FIGURE 3.8

A boundary representation model of a solid object. (a) Solid model. (b) Boundary surfaces.

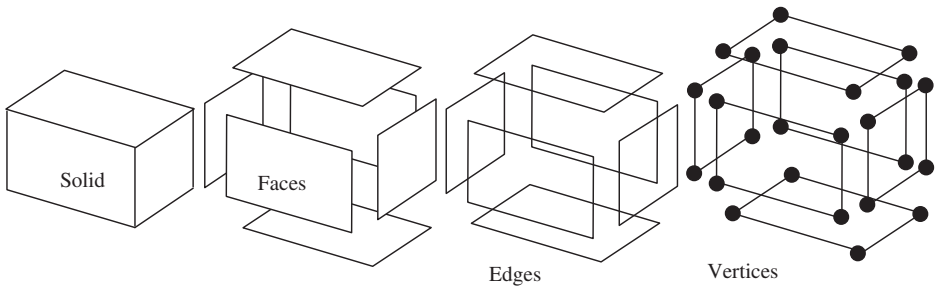


FIGURE 3.9

The boundary representation model of a rectangular block.

joined along the common edges of their respective neighboring faces. Every edge—a straight line in this case—is bounded by its end points.

One advantage of the B-rep model is that the model is fully evaluated; that is, all geometric entities are explicitly defined and are ready for display. However, a complex B-rep model requires a relatively large database. It is less intuitive to create solid models using the B-rep method as compared with CSG because users must deal with points, curves, and surfaces instead of primitives that are much more relevant to physical objects. More importantly, a B-rep model constructed by a designer may be invalid, and a B-rep model must be verified for its topology before putting it to use.

How can we (or a computer, in this case) tell if a B-rep model is topologically valid? In a topologically valid model, all faces are properly "glued" to wrap the solid object airtight, all edges are properly "joined" that fence the face, and all edges are properly bounded by end vertices. In addition, there must be no dangling faces or edges, and no split solid object.

Which objects in Figure 3.10 are topologically valid? Apparently, objects in Figures 3.10(a) and (b) are valid, but not Figure 3.10(c). It is apparent that normal objects found in nature have the property that, at every point on the boundary, a small enough sphere around the point is divided into two pieces: one inside and one outside the object. This property can be easily verified for the objects in Figures 3.10(a) and (b). The so-called nonmanifold models break this rule. For example, in the object shown in Figure 3.10(c), a small sphere around any point on the four edges of the rectangle at the bottom face of the cube is divided into four pieces: two inside and two outside. Essentially, the object in Figure 3.10(c)

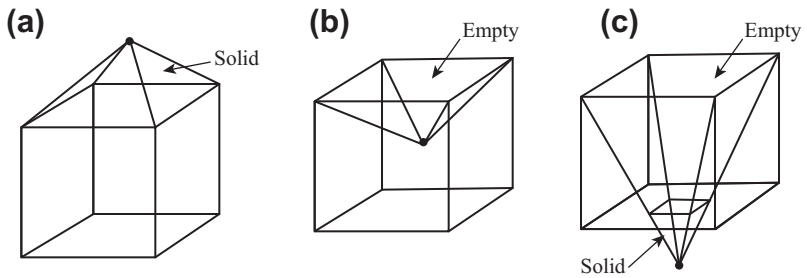


FIGURE 3.10

Examples of manifold (a, b) and nonmanifold (c) objects.

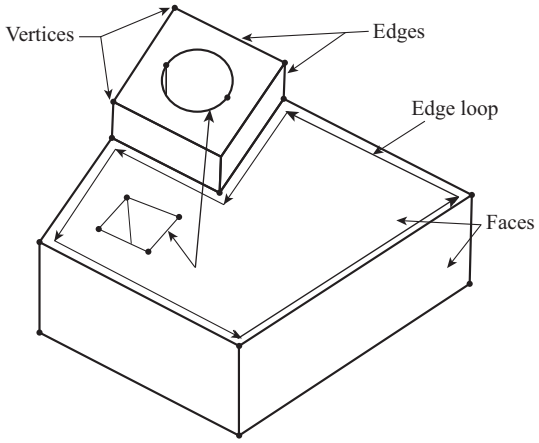


FIGURE 3.11

Illustration for the topological entities employed in the Euler-Poincare law.

is two solids "welded" along the edges of the rectangle on the bottom face. The welded edges are infinitely thin without a cross-sectional area, which is physically impossible. A nonmanifold object, such as the one in Figure 3.10(c), is considered to be topologically invalid.

To ensure the topological validity of a B-rep model, the number of its topological entities must satisfy the Euler-Poincare law, which is stated as follows:

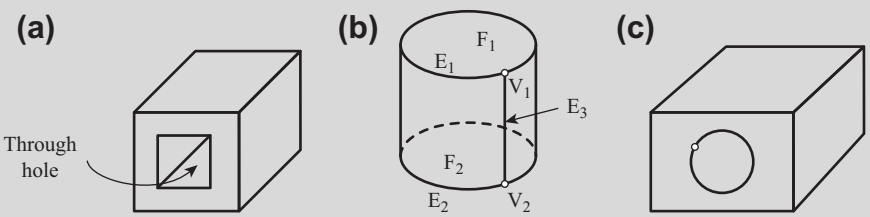
$$v - e + f = 2(s - h) + r \tag{3.3}$$

where v, e, f, s, h, and r are the numbers of vertices, edges, faces, solids, through holes, and rings, respectively. Note that r can be a ring or an inner loop of edges that are completely within a face. Figure 3.11 offers an illustration for the topological entities mentioned in Eq. 3.3.

The following simple examples illustrate the law and verify the topological validity of the respective physical objects.

EXAMPLE 3.1

Use the Euler-Poincare law to verify if the following objects are topologically valid: (a) a rectangular block with a rectangular through hole, (b) a circular cylinder, and (c) a rectangular block with a circular through hole.



Continued

EXAMPLE 3.1—cont'd

Solutions

For part (a), we have $f = 10$ (four exterior, four interior, and two ends), $v = 16$, $e = 24$, $s = 1$, $h = 1$, and $r = 2$. From Eq. 3.3, we have

$$v - e + f = 16 - 24 + 10 = 2$$

and

$$2(s - h) + r = 2(1 - 1) + 2 = 2.$$

Therefore, part (a) is topologically valid. Physically, the block is a valid object.

For part (b), we have $f = 3$, $v = 2$, $e = 3$, and $s = 1$. Note that a silhouette edge must be added (and the associated vertices) to a cylindrical surface when counting the number of topological entities. From Eq. 3.3, we have

$$v - e + f = 2 - 3 + 3 = 2$$

and

$$2(s - h) + r = 2(1 - 0) + 0 = 2.$$

Therefore, part (b) is topologically valid.

We are applying the same principle for part (c)—that is, adding a silhouette edge to the circular cylindrical surface inside the rectangular block.

Hence, for part (c), we have $f = 6 + 1 = 7$, $v = 8 + 2 = 10$, $e = 12 + 3 = 15$, $s = 1$, $h = 1$, and $r = 2$. From Eq. 3.3, we have

$$v - e + f = 10 - 15 + 7 = 2$$

and

$$2(s - h) + r = 2(1 - 1) + 2 = 2.$$

Therefore, part (c) is topologically valid.

Boundary representation is essentially a local representation connecting faces, edges, and vertices. An extension of this is to group the primitive geometric entities of the shape into logical units called geometric features. Features are the basis of many other developments, allowing high-level "geometric reasoning" about shape for comparison, process planning, manufacturing, etc. Feature-based modeling is discussed next in Section 3.3.

Compared to the CSG representation, which uses only primitive objects and Boolean operations to combine them, boundary representation is more flexible and has a much richer operation set. This makes boundary representation a more appropriate choice for CAD systems. CSG was used initially by several commercial systems because it was easier to implement. The advent of reliable commercial B-rep kernel systems, such as Parasolid and ACIS, has led to widespread adoption of B-rep for CAD. B-rep kernel systems offer CAD-like operations, such as protrusion, chamfer, blending, drafting, shelling, tweaking, and other operations. Note that most CAD systems employ both CSG and B-rep for solid modeling, or at least the major principles of these methods. In general, CSG keeps the relationship between features, and B-rep stores topological and geometric data for display and computations. More about geometric modeling kernels can be seen in Section 3.3.7.

3.3 FEATURE-BASED PARAMETRIC SOLID MODELING

Most modern CAD software employs feature-based parametric solid modeling as the major interface for users to interactively create solid models. Feature-based modeling approach is more desirable in constructing solid models, in which designers use features that correspond to physical entities to construct solid models, instead of dealing with primitive geometric entities, such as points, curves, and solid primitives. The features available in CAD are usually designed to relate to how engineers think in their design and manufacturing work. The parametric modeling method allows designers to create solid models in such a way that by varying a few parameters (e.g., geometric dimensions), the solid models rebuild automatically as intended (i.e., capturing design intent). For example, a hole in the block shown in Figure 3.12(a) is intended to stay at the middle of the block when the width of the block changes. To capture this design intent, first the sketch profile of the base block must be fully defined (Figure 3.12(b)), with dimension $d2$ as a design variable that is to vary. The hole must be placed to the profile of the base block with its width dimension ($d1$ in Figure 3.12(c)) related to that of block width $d2$, as $d1 = 0.5d2$. This one-way parameter assignment is called parametric modeling.

In most CAD, the base block is a protrusion feature generated by extruding the sketch profile shown in Figure 3.12(b) along an extrusion direction that is normal to the sketch profile. The hole is an extrude cut feature that does not require users to sketch its profile, which is called a pick-and-place feature.

The question is how CAD software captures our design intent. How does CAD handle the sketch profile of the block when you make a change? How does CAD rebuild the part and what are the potential pitfalls you, as a designer, should avoid?

In this section, we intend to answer these questions. We start by revisiting "features" to have a better understanding of the terminology so that the "feature-based modeling" becomes more vivid. We then discuss the variational modeling method that CAD employs to support the needed calculation for determining a sketch profile. We discuss parent–child relationships, which are generated when features

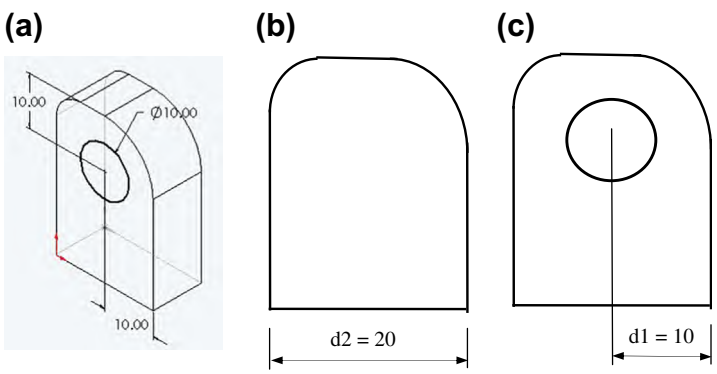


FIGURE 3.12

(a) The block example for the illustration of design intent capturing using feature-based parametric solid modeling method. (b) Sketch profile of the base block with width dimension $d2$ shown. (c) Position of the hole by dimension $d1$, which can be parametrically related to dimension $d2$.

are added to the solid model. Finally, we discuss the parametric modeling method and the solid modeling procedure we often exercise in CAD, in which we show how CAD rebuilds a part by walking through the steps using a simple example. We also discuss a newly developed modeling method called direct modeling. This section is wrapped up by a short introduction to geometric modeling kernels.

3.3.1 GEOMETRIC FEATURES

The term "feature" implies different meanings in different engineering disciplines. This has resulted in many ambiguous definitions for feature. A feature in computer-aided design (CAD) usually refers to a region of a part with certain geometric or topological properties (Pratt and Wilson, 1985). These are more precisely called geometric (or form) features. Geometric features contain both shape information and parametric information of a region of interest. They are now ubiquitous in most current CAD software, where they are used as the primary means of creating 3D solid models.

Another frequently used feature is the manufacturing feature, which can be defined simply as a geometric shape and its manufacturing information to create the shape. Manufacturing features support the generation of process plans in a feature-based process planning system. Machining features are an important subset of manufacturing features. A machining feature can be regarded as the volume swept by a "cutting" tool, which is always a negative (subtracted) volume. Some CAM software, such as CAMWorks, offers an automatic (manufacturing) feature recognition capability that recognizes manufacturing features embedded in CAD solid models and generates a toolpath accordingly. There are also tolerance features that specify deviations from the nominal form (or shape), size, or locations, such as surface flatness, circularity, and concentricity. Finally, there is the concept of assembly feature, which encodes the assembly method between connected components.

The features mentioned previously are highly related to part geometry. There are also non-geometric features, such as material features that specify material composition and heat treatment for a part.

In this subsection, we discuss geometric features from a design perspective. Instead of providing a generic and philosophical discussion, we narrow the focus to CAD solid modeling. Nowadays, almost everything that contributes to the construction of a solid model in CAD is called a feature. Basically, geometric features involved in creating a solid object can be categorized into the five groups: construction (or datum) features, shape (or protrusion) features, pick-and-place (or hard-coded) features, mirror and pattern features, and thickened features, as shown in Figure 3.13. Therefore, features can be thought of as the individual shapes that, when combined, make up the part.

Construction or datum features—such as coordinate systems, planes, axes, or points—are auxiliary entities that aid solid model creation. Default construction features, including a coordinate system and

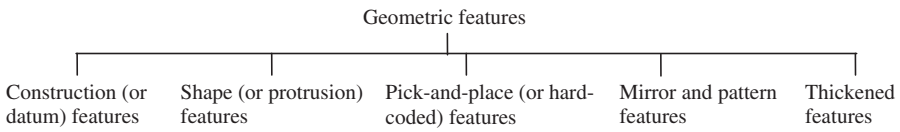


FIGURE 3.13

Classification of geometric features.

three perpendicular planes (front, top, and right), are provided in CAD as the starting point for part solid modeling (and assembly).

Protrusion features (some also call them "sweep" features) are the most important set of features that support solid modeling. Such features include extrude, sweep, loft (or blend), and revolve, in which sketch profiles are required, as illustrated in Figure 3.14. In addition to protrusion that adds volume, protrusion can also be used to create cut features that remove volume from existing objects. Attributes, such as protrusion direction (one or both sides), are options offered to designers to complete a protrusion feature conveniently.

Pick-and-place features are hard-coded features, including chamfer, fillets, rounds, draft, and holes, which are placed on a face or an edge of existing objects without sketching a profile. Such features are often added in the final stage of the solid modeling process.

Mirror and pattern features are created from existing features, as illustrated in Figure 3.15. Mirror copies the selected features or all features, mirroring them about the selected plane or face (Figure 3.15(c)). Pattern, as shown in Figures 3.15(a) and (b), repeats the selected features in an array based on a seed feature. The array can be linear (a linear pattern), a circular (a circular pattern), or following a curve. Some CAD, such as SolidWorks, offers feature copy and paste capabilities, in which designers can pick an existing feature (e.g., a through hole), then copy and paste it on a different face of the solid object in a different orientation.

The thickened feature creates a solid feature by thickening one or more adjacent surfaces. For example, the tracked vehicle roadarm surface model discussed in Chapter 2 (Section 2.6.2) was created in B-spline surfaces, imported into SolidWorks, and then thickened for a solid model in support of structural analysis using FEA. Note that except for the construction features, geometric features are solid features.

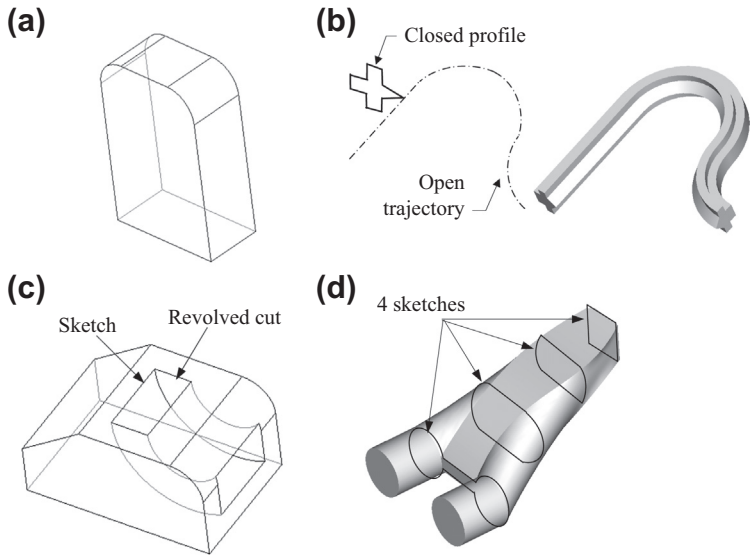


FIGURE 3.14

Protrusion features. (a) Extrusion feature. (b) Sweep feature. (c) Revolved cut feature. (d) Blend feature.

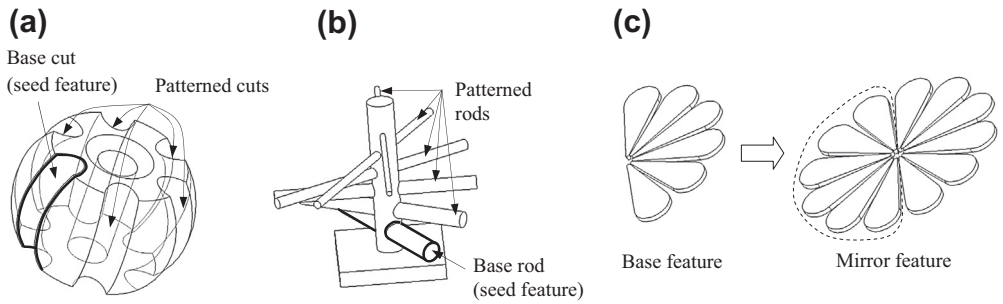


FIGURE 3.15

Copy, pattern, and mirror features. (a) Patterned cut features. (b) Patterned extrusion features. (c) Mirror feature.

As mentioned earlier, protrusions are probably the most important ones in supporting designers to create most features in solid modeling. Protrusions features (including cuts) require designers to sketch a profile, in which a section view of the feature is defined. How does CAD support designers to create sketch profiles interactively? How does CAD receive inputs from the designer, then formulate and solve equations to define the sketch profile mathematically? We discuss sketch profiles in the next subsection.

3.3.2 SKETCH PROFILES

When we start a new part, we pick a sketch plane and create a profile. The profile is the basis for a 3D model. We usually create a profile on one of the default construction planes (front, top, and right), or a created plane. In sketching a profile, as a CAD user, we create an open or closed profile with lines, arcs, and so on. A CAD system, such as SolidWorks or Pro/ENGINEER, automatically adds sketch relations (also called sketch constraints) to relate or constrain entities; for example, a straight line connects to a circular arc with a tangent relation at the junction point. After completing the profile, we add dimensions and enter proper values to adjust the profile that meets our design requirements. After entering or modifying a dimension value, CAD is able to adjust the profile as a logical consequence of the change. How does the CAD system adjust the sketch profile? CAD employs the so-called variational modeling theory in sketch mode.

3.3.2.1 Sketch Relations

In this section, we use examples to illustrate the sketch relations and variational modeling technique. We assume SolidWorks sketch mode to be more specific. Other CAD systems follow a similar approach.

We often start the sketch profile at the origin; that is, we use the origin as the anchor for the profile. SolidWorks (and other modern CAD systems) creates relations for the geometric entities in the profile, based on how these entities are created. Designers add dimensions (and relations between dimensions) to make the profile fully defined (or fully constrained).

In SolidWorks, before creating any dimensions, the geometric entities, including lines and vertices, are either in black or (mostly) in blue color. As illustrated in Figure 3.16 with a simple example, black

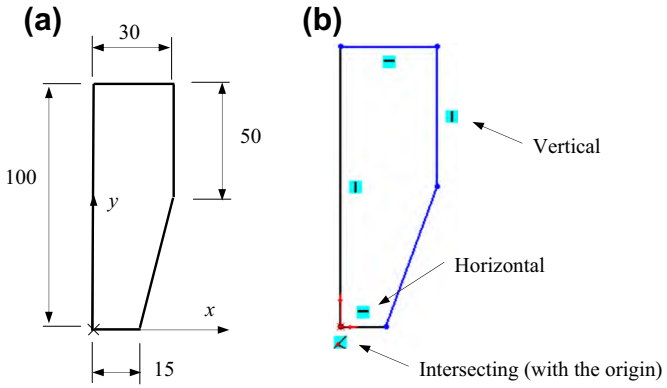


FIGURE 3.16

Sketch profile as designed (a) and with underconstrained relations (b).

indicates that the entity is fully defined. Blue indicates that the entity is not fully defined and is free to change in a certain way. You may drag a vertex or line in blue color to see how it can be changed in SolidWorks.

When you add dimensions or relations, affected entities will change from blue to black color, indicating they become fully defined as a result. When all the entities of the sketch are in black, the entire sketch is fully defined. In the model tree (called Browser in SolidWorks), the (−) sign in front of the sketch node is removed. Sometimes, the sketch is overconstrained when a conflict occurs or more dimensions (or relations) are created than required.

Note that some CAD systems, such as Pro/ENGINEER, offer "smart" sketching tools. When turned on, design intent is inferred, and sketch relations and dimensions are added automatically to make the sketch profile fully defined. In some cases, line and curve entities are slightly adjusted with imposed relations. For example, two straight lines that are nearly perpendicular may "snap" perpendicular with a perpendicularity relation (Figure 3.17(a)), and two fillets with about the same radii may be adjusted to have equal radii with an equal radii relation added (Figure 3.17(b)). For more details on the sketch relations in SolidWorks and Pro/ENGINEER, please refer to Appendix 3A.

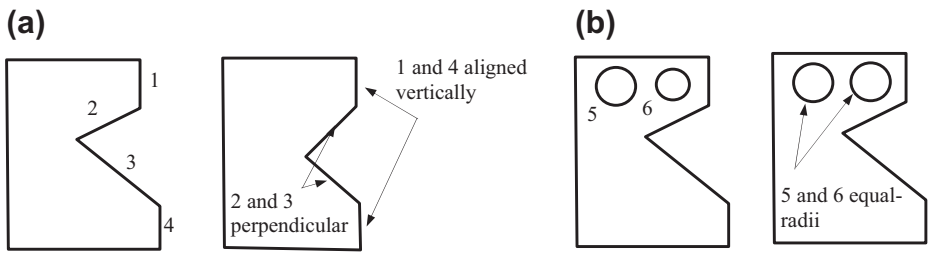


FIGURE 3.17

Effects of imposed sketch relations to the profile. (a) Perpendicular and alignment relations were added. (b) Equal-radii relation was added.

The question is how SolidWorks knows what dimensions and relations are just right for the sketch (i.e., to make it fully defined). How does SolidWorks figure out if your sketch is overdefined or underdefined?

3.3.2.2 Variational Modeling

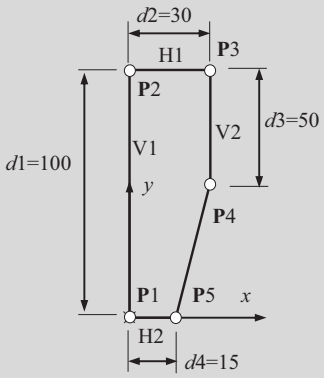
CAD software employs the variational modeling theory in sketch mode. The first step involved in variational modeling is identifying a set of characteristic points (or vertices) on the sketch profile. For example, the sketch shown in Figure 3.16 has five characteristic points that are at the corners of the polygon. Then, a system of equations is derived that incorporates the relations and dimensions defined on the sketch to relate the x- and y-locations of these characteristic points. The system of equations is solved to determine the locations of the characteristic points.

In formulating the system of equations, the number of equations must be identical to the number of unknowns and equations must be linearly independent for a unique solution. This is when a sketch is called fully defined. When the dimension values are changed, the same system of equations is solved again for the locations of the characteristic points.

When the number of equations is greater than the number of unknowns, we have an overdefined sketch. When the number of equations is less than the number of unknowns, we have an underdefined case, where a (−) sign will stay in front of the sketch in the Browser.

EXAMPLE 3.2

Determine if the following sketch profile with the relations and dimensions is fully defined. In this sketch, **P**1 is fixed to the origin, and there are four relations (two horizontal and two vertical) and four dimensions. If we change the dimension *d*3 from 50 to 100 and *d*4 from 15 to 30, what will happen to the profile? Would CAD accept such changes and be able to regenerate the profile? If we add a dimension, such as the length of the line segment **P**4**P**5, would the sketch profile still be fully defined?



Solutions

There are five characteristic points in the sketch profile, **P**1, **P**2, **P**3, **P**4, and **P**5; therefore, we have $5 \times 2 = 10$ unknowns and we need to have 10 linearly independent equations to solve for the unknowns. The following equations are derived from the relations and dimensions defined in the sketch.

EXAMPLE 3.2—cont'd

Because point **P**1 coincides with the origin, we have the following two equations:

$$\text{Coincident:} \quad \mathbf{P}1x = 0 \tag{1}$$

and

$$\text{Coincident:} \quad \mathbf{P}1y = 0 \tag{2}$$

The remaining equations are:

$$\text{V1:} \quad \mathbf{P}2x - \mathbf{P}1x = 0 \tag{3}$$

$$d1: \quad \mathbf{P}2y - \mathbf{P}1y = d1 \tag{4}$$

$$\text{H1:} \quad \mathbf{P}3y - \mathbf{P}2y = 0 \tag{5}$$

$$d2: \quad \mathbf{P}3x - \mathbf{P}2x = d2 \tag{6}$$

$$\text{V2:} \quad \mathbf{P}4x - \mathbf{P}3x = 0 \tag{7}$$

$$d3: \quad \mathbf{P}3y - \mathbf{P}4y = \text{d3} \tag{8}$$

$$\text{H2:} \quad \mathbf{P}5y - \mathbf{P}1y = 0 \tag{9}$$

$$d4: \quad \mathbf{P}5x - \mathbf{P}1x = d4 \tag{10}$$

These ten equations are linearly independent. How can you tell? You may arrange these ten equations into a matrix form and check the rank of the matrix.

If we change the dimension $d3$ from 50 to 100 and $d4$ from 15 to 30, points **P**4 and **P**5 coincide, making the length of the line segment **P**4**P**5 zero. Most CAD software, such as SolidWorks, will not accept any line or curve entity with zero length.

Because we have already ten linearly independent equations that solve uniquely for the ten unknowns, adding more dimensions, such as the length dimension for the line segment **P**4**P**5, causes the profile to become overdefined.

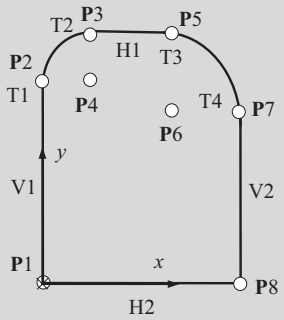
Now, let us take a look at a slightly more complex problem, in which the profile consists of circular arcs.

EXAMPLE 3.3

Add adequate dimensions to the following sketch profile with the given relations to make the profile fully defined. In this sketch, **P**1 is fixed to the origin, and there are another eight relations, as shown below. Formulate system of equations and solve them for the locations of the characteristic points that determine the shape of the profile.

Continued

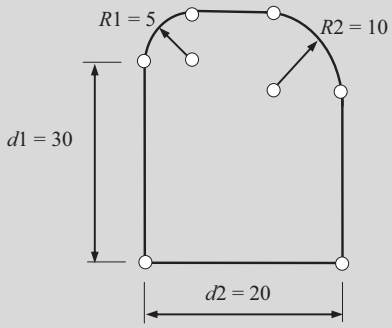
EXAMPLE 3.3—cont'd



Tangent (T1)
Tangent (T2)
Tangent (T3)
Tangent (T4)
Vertical (V1)
Horizontal (H1)
Vertical (V2)
Horizontal (H2)

Solutions

There are eight characteristic points (including the arc centers); therefore, there are $8 \times 2 = 16$ unknowns, and we need to have 16 linearly independent equations to solve for the unknowns. The following equations are derived from the relations given to the profile shown above.



$$\text{Fix:} \quad \mathbf{P}1x = 0, \tag{1}$$

and

$$\text{Fix:} \quad \mathbf{P}1y = 0 \tag{2}$$

$$\text{V1:} \quad \mathbf{P}2x - \mathbf{P}1x = 0 \tag{3}$$

$$\text{T1 (and V1):} \quad \mathbf{P}4y - \mathbf{P}2y = 0 \tag{4}$$

$$\text{T2:} \quad \mathbf{P}3x - \mathbf{P}4x = 0 \tag{5}$$

$$\text{H1:} \quad \mathbf{P}5y - \mathbf{P}3y = 0 \tag{6}$$

$$\text{T3 (and H1):} \quad \mathbf{P}5x - \mathbf{P}6x = 0 \tag{7}$$

$$\text{T4:} \quad \mathbf{P}7y - \mathbf{P}6y = 0 \tag{8}$$

EXAMPLE 3.3—cont'd

$$\text{V2:} \quad \mathbf{P}7x - \mathbf{P}8x = 0 \tag{9}$$

$$\text{H2:} \quad \mathbf{P}1y - \mathbf{P}8y = 0 \tag{10}$$

We need six more equations. These six additional equations come from the dimensions we are about to add to the profile.

If you add the four dimensions $d1$, $d2$, $R1$, and $R2$, as shown on the previous page, the profile becomes fully defined. Why? Let us take a look at the equations that the dimensions will provide.

$$d1: \quad \mathbf{P}2y - \mathbf{P}1y = d1 \tag{11}$$

$$R1 \text{ (T1 and T2):} \quad \mathbf{P}4x - \mathbf{P}2x = R1 \tag{12}$$

$$\mathbf{P}3y - \mathbf{P}4y = R1 \tag{13}$$

$$R2 \text{ (T1 and T2):} \quad \mathbf{P}5y - \mathbf{P}6y = R2 \tag{14}$$

$$\mathbf{P}7x - \mathbf{P}6x = R2 \tag{15}$$

$$d2: \mathbf{P}8x - \mathbf{P}1x = d2 \tag{16}$$

Now we have all 16 equations identified. Among them, Eqs 1—3, and 10 are trivial; therefore, they are removed together with the four unknowns (i.e., $\mathrm{P}1x = \mathrm{P}1y = \mathrm{P}2x = \mathrm{P}8y = 0$).

We assemble the remaining 12 equations for the 12 unknowns in a matrix form as follows.

$$\begin{array}{l}(11)\\(5)\\(13)\\(12)\\(4)\\(7)\\(6)\\(15)\\(14)\\(9)\\(8)\\(16)\end{array}\begin{bmatrix}1&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&-1&0&0&0&0&0&0&0&0\\0&0&1&0&-1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&0&0&0\\-1&0&0&0&1&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&-1&0&0&0&0\\0&0&-1&0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&0&1&0&-1&0&0\\0&0&0&0&0&0&-1&0&1&0&0&0\\0&0&0&0&0&0&0&0&0&1&0&-1\\0&0&0&0&0&0&0&0&-1&0&1&0\\0&0&0&0&0&0&0&0&0&0&0&1\end{bmatrix}\begin{bmatrix}\mathbf{P}2y\\\mathbf{P}3x\\\mathbf{P}3y\\\mathbf{P}4x\\\mathbf{P}4y\\\mathbf{P}5x\\\mathbf{P}5y\\\mathbf{P}6x\\\mathbf{P}6y\\\mathbf{P}7x\\\mathbf{P}7y\\\mathbf{P}8x\end{bmatrix}=\begin{bmatrix}d1\\0\\R1\\R1\\0\\0\\0\\-R2\\-R2\\0\\0\\d2\end{bmatrix}=\begin{bmatrix}30\\0\\5\\5\\0\\0\\0\\-10\\-10\\0\\0\\20\end{bmatrix}$$

The system of equation can be solved using, for example, Matlab. We are solving five cases: Case 1: $d1 = 30$, $R1 = 5$, $R2 = 10$, $d2 = 20$ (base case); Case 2: $d1 = 60$, $R1 = 5$, $R2 = 10$, $d2 = 20$ (taller profile); Case 3: $d1 = 30$, $R1 = 5$, $R2 = 5$, $d2 = 20$ (equal fillet radii); Case 4: $d1 = 30$, $R1 = 5$, $R2 = 15$, $d2 = 20$ (zero length profile); and Case 5: $d1 = 30$, $R1 = 5$, $R2 = 25$, $d2 = 20$ (penetrating profile).

The Matlab script for solving the equations is shown (next page), followed by the resulting profiles.

Continued

EXAMPLE 3.3—cont'd

EDU≫ $a =$

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

Case 1

EDU» *c*=[30
0
5
5
0
0
0
-10
-10
0
0
20]

EDU» *b*=inv(*a*)*c

b =
30
5
35
5
30
10
35
10
25
20
25
20

*d*1
0
*R*1
*R*1
0
0
0
-*R*2
-*R*2
0
0
*d*2

b =
p2*y*
p3*x*
p3*y*
p4*x*
p4*y*
p5*x*
p5*y*
p6*x*
p6*y*
p7*x*
p7*y*
p8*x*

Case 2

EDU» *c*=[*60*
0
5
5
0
0
0
-10
-10
0
0
20]

EDU» *b*=inv(*a*)*c

b =
60
5
65
5
60
10
65
10
55
20
55
20

Case 3

EDU» *c*=[30
0
5
5
0
0
0
-5
-5
0
0
20]

EDU» *b*=inv(*a*)*c

b =
30
5
35
5
30
15
35
15
30
20
30
20

Case 4

EDU» *c*=[30
0
5
5
0
0
0
-15
-15
0
0
20]

EDU» *b*=inv(*a*)*c

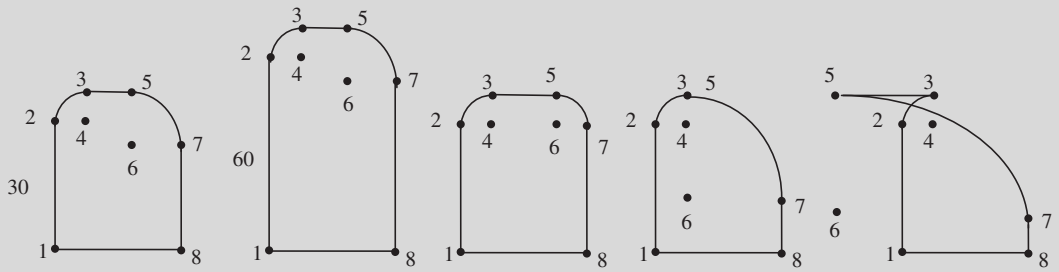
b =
30
5
35
5
30
5
35
5
20
20
20
20

Case 5

EDU» *c*=[30
0
5
5
0
0
0
-25
-25
0
0
20]

EDU» *b*=inv(*a*)*c

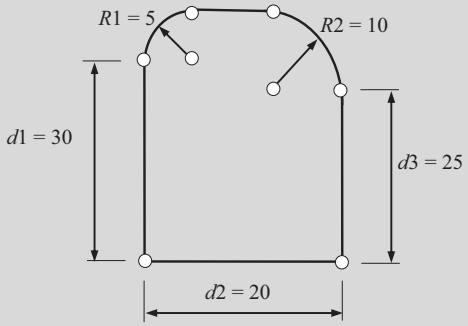
b =
30
5
35
5
30
-5
35
-5
10
20
10
20



EXAMPLE 3.3—cont'd

A number of important points are observed from Example 3.3:

1. Solutions exist for all five cases, which is because the number of equations and number of unknowns are identical, and the equations are linearly independent. As a result, the matrix equation is nonsingular and can be solved for a unique solution.
2. It is apparent that if we add a dimension $d3$ shown below, the sketch becomes overconstrained. Now, if we add $d3$ and remove $d2$, would the sketch be fully defined? The answer is no. Adding $d3$ creates a reddendum dimension because $d3$ is determined by $d1 + R1 - R2$. Removing the equations contributed by $d2$ and adding those contributed by $d3$ will make the system of equations linearly dependent; therefore, the resulting matrix equation becomes singular.



3. Cases 1—3 are regular profiles that CAD generates. For Case 4, because there is a zero-length line segment **P3P5**, some CAD programs, such as SolidWorks, prompt an error message of zero-length line entity and will not generate the profile. For Case 5, although the entities penetrate to each other, indicating a physically infeasible profile, some CAD systems, such as Pro/ENGINEER, check the validity of the profile and prompt with a warning message; however, others, such as SolidWorks, do not catch the problem and generate an invalid sketch profile anyway.
4. In both examples, we have all linear equations. In some cases, for instance, if angle dimensions are present, nonlinear equations are required. In these cases, an iterative numerical method, such as Newton's method, may be employed for solving the nonlinear equations.

3.3.3 PARENT–CHILD RELATIONSHIPS

Once a sketch profile is completed, a solid feature can be created by using, for example, one of the protrusion capabilities. The first solid feature serves as the first building block, called the base feature, in the model construction process. Follow-on features are added to the base feature or existing solid features.

Depending on the sequence of feature construction, there is a parent–child relationship created between solid features. For example, the part shown in Figure 3.18(a) consists of five solid features. The first solid feature is the base block, which was created by extruding a sketch profile like that of Figure 3.12(b) along the extrusion direction perpendicular to the sketch plane of the profile. A center through hole was then added as a cut extrusion feature (or a pick-and-place feature), with a sketch placed on the front face of the base block, in which the hole is placed with position dimensions referred to the right and top edges of the base block (see Figure 3.12a), respectively. As a result,

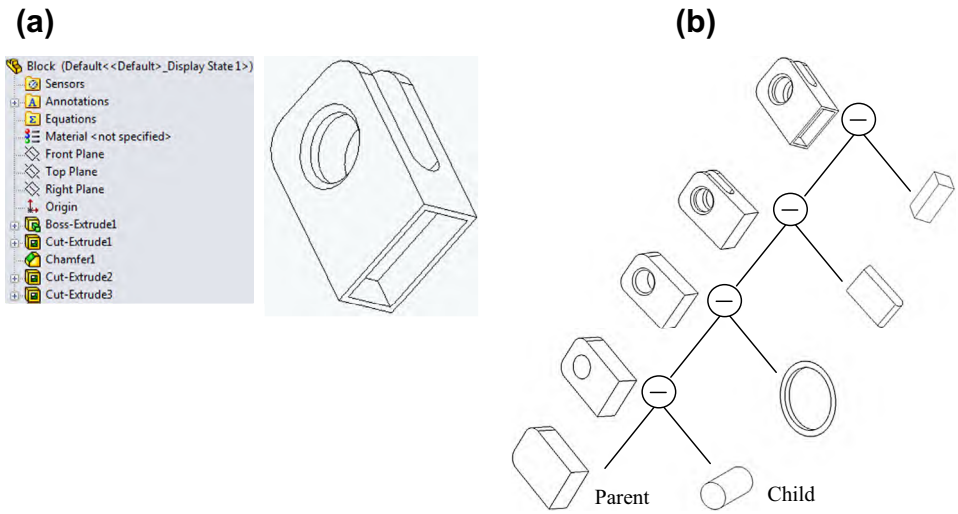


FIGURE 3.18

The parent–child relationships. (a) The block solid model and feature tree in browser. (b) Feature construction sequence in a constructive solid geometry-like tree.

hole became a child feature of the base block, as illustrated in Figure 3.18(b). The third feature is a chamfer (a pick-and-place feature) placed on the outside circle of the hole on the front face of the block. As a result, the chamfer is a child feature of the hole. The fourth and fifth features are the side cut and bottom slot, respectively, as shown in Figure 3.18(b); both are child features of the base block.

In addition to parent–child relationships between solid features, a solid feature may be a child feature of a construction feature, and vice versa. For example, the base block of Figure 3.18(b) is a child feature of a default construction plane because its sketch profile was created on the plane.

Parent–child relationships are critical in the construction and rebuild of the solid model. Changes made to the parent feature will propagate to all child features during the rebuild. On some occasions, the part may not be rebuilt successfully due to numerous reasons, mainly because the part is not properly parameterized. Also, operations such as deleting, suppressing, or hiding a parent feature will affect all its child features. Therefore, it is extremely critical that designers arrange the sequence of feature construction and the way the child feature is related to its parent feature. A desirable solid model should have less coupled parent–child relationships between features. Solid model construction sequence or history is critical in the feature-based modeling approach.

3.3.4 PARAMETRIC MODELING

Unlike the variational modeling technique that formulates and solves a system of equations, parametric modeling adopts a one-way assignment approach. For example, two dimensions $d0$ and $d1$ can be parametrically related as $d0 = d1 \times 2$, in which d0 is a dependent parameter and $d1$ is an independent parameter that is free to change.

Such "assignment"-type equations are explicit and they are solved sequentially, in which each assigned value is computed as a function of previously assigned or computed values. Therefore, parametric modeling is also called "unidirectional" modeling or "procedural" modeling. In the above example, $d1$ must be defined first, and then d0 can be evaluated. The solid model must be rebuilt (regenerated) by propagating the changed parameter (dimension) through all equations that involve the parameter.

Computation in solving the explicit equation sequentially is efficient and straightforward. However, this approach lacks flexibility in relating parameters. For example, in the variational modeling method, $d0 = d1 \times 2$ can be written as $d0 - d1 \times 2 = 0$, in which either $d0$ or $d1$ can be independent.

3.3.5 SOLID MODELING PROCEDURE IN CAD

After reviewing the solid modeling methods discussed so far, we revisit the general process of creating solid models in CAD. In the meantime, we walk through how CAD rebuilds the part when we make a design change, using a simple example. By going through this exercise, we hope you gain a better understanding on the behind-the-scenes operations while using CAD for solid modeling.

In general, as illustrated in Figure 3.19, when we start a new solid model, we are given datum features, such as datum planes and datum coordinate systems. It is in general a good idea to develop a modeling plan before beginning the actual modeling work. More about modeling plans is discussed in Section 3.4. At the beginning of creating a new part, we usually pick a datum plane and create a sketch profile for the first (or base) solid feature using one of the protrusion capabilities. As discussed before, a variational modeling technique is employed in CAD to determine the locations of characteristic points of the sketch profile. After the base solid feature is created, we either add more solid features by repeating the same process, sketch and make a cut feature, or place a pick-and-place feature on the existing features. We repeat some or all steps to create more features. In the meantime, CAD records the feature creation sequence in the model tree and parent–child relationships between features. Once a solid model is completely created, we often make a few adjustments to make sure the solid model accurately represents the design of the part. CAD rebuilds the part based on the changes we made by updating features (both datum and solid features) following the feature creation sequence, one feature at a time. For each feature, the CAD system does the following:

1. Takes the new dimension values, from user input and computation through parameter relations, to update the sketch profile first. In the sketch profile, a variational modeling technique is exercised. The system of equations that govern the profile of the section are solved again for the new parameter values.
2. Rebuilds the geometric features using the new sketch profile by
   a. New parameter values in the protrusion direction, including extrude, revolve, sweep, or loft; and
   b. Feature attributes, such as one side or both sides.
3. If the feature cannot be rebuilt (e.g., when an invalid geometric feature is encountered), an error message will appear. For example, in SolidWorks, a rebuild error window appears, as shown in Figure 3.20.

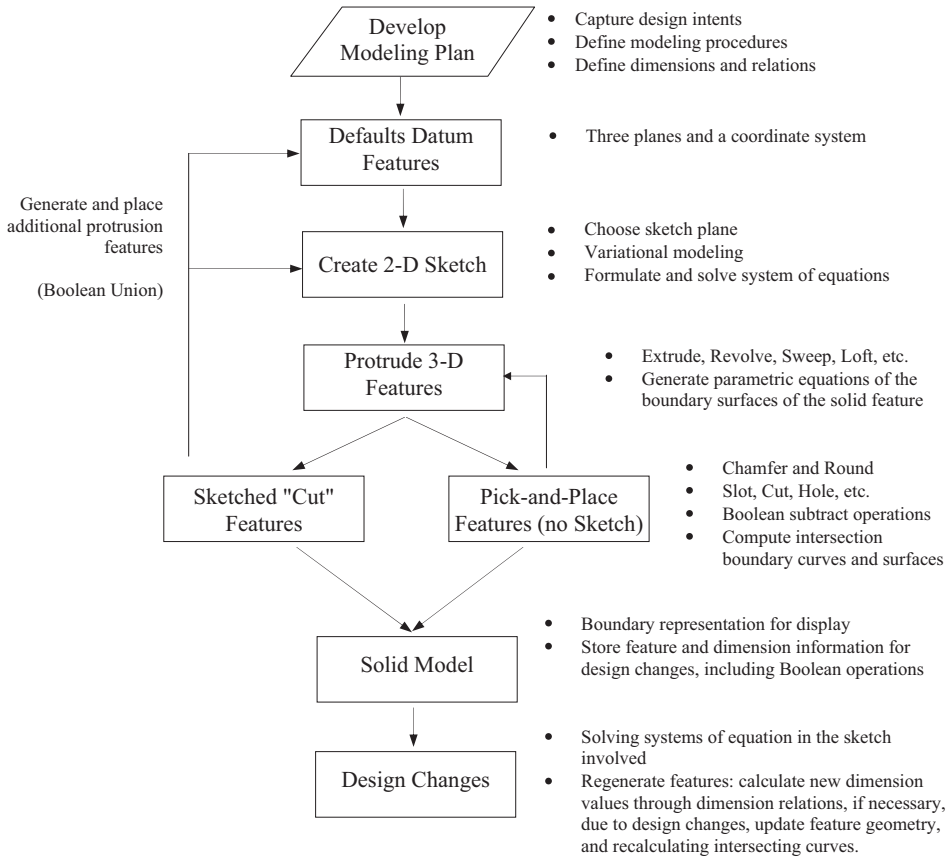


FIGURE 3.19

General solid model creation process.

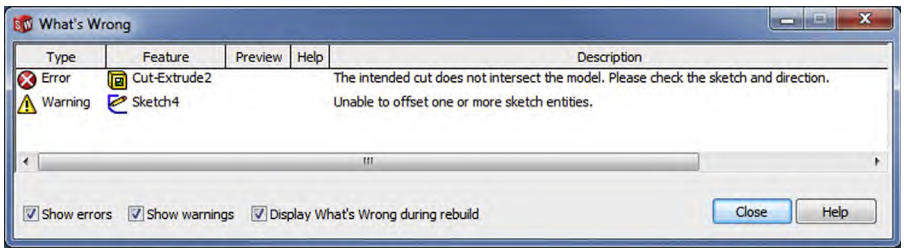


FIGURE 3.20

The rebuild error message window in SolidWorks.

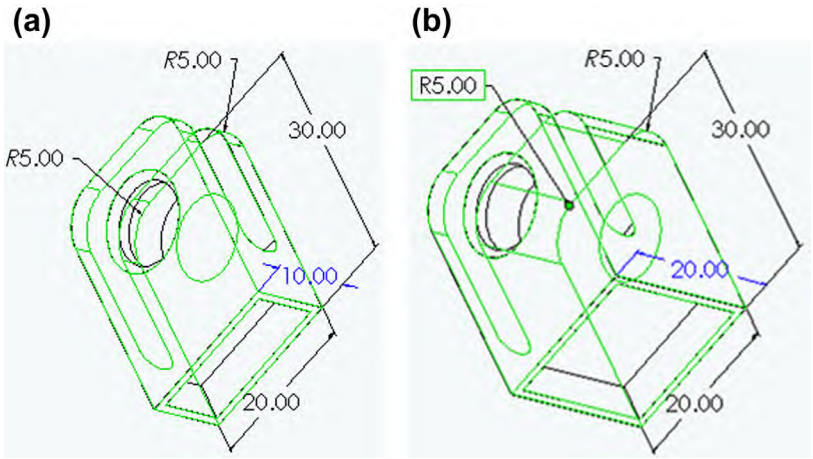


FIGURE 3.21

Change of the depth dimension of the base block from 10 to 20. (a) Solid model before design change. (b) Solid model after design change.

4. When this error message appears, we must read the messages carefully and identify which feature is problematic. Once the problematic feature is identified, it is good practice to try to figure out what the problem was, and then undo the change to restore the dimension values, fix the problem, and try a similar change again. In SolidWorks or Pro/ENGINEER, you may choose *Edit* > *Undo Change Dim* (or CTL + Z) to undo the change.
5. Find child features through the parent–child relations. Note that while rebuilding child features, intersecting curves of feature boundary surfaces may need to be recomputed.
6. Repeat steps 1–5 until all the features are rebuilt.

In the following, we walk through the part rebuild process using the block example shown in Figure 3.18. We first change the depth dimension of the base block ($d5$) from 10 to 20, as shown in Figure 3.21. The regeneration follows the feature creation sequence, as shown in Figure 3.18(b), which is described below.

1. First feature: base block
   The sketch profile is unchanged. There is no need to resolve the system of equations. Therefore, the only action for CAD is to update the width of the base block on one side along the extrude direction, which is the attribute of the base block.
2. Second feature: big hole
   Both placement data (placement plane, placement references, and dimensions) and hole dimension are unchanged. The hole is rebuilt following the feature attribute (i.e., through all and one side). The intersecting curve (in this case, the circle in the back face) is computed.
3. Third feature: side cut
   Check sketch profile. Is the sketch profile changed? Why and why not? The cut feature is regenerated following the feature attribute (i.e., through all and cut directions), as

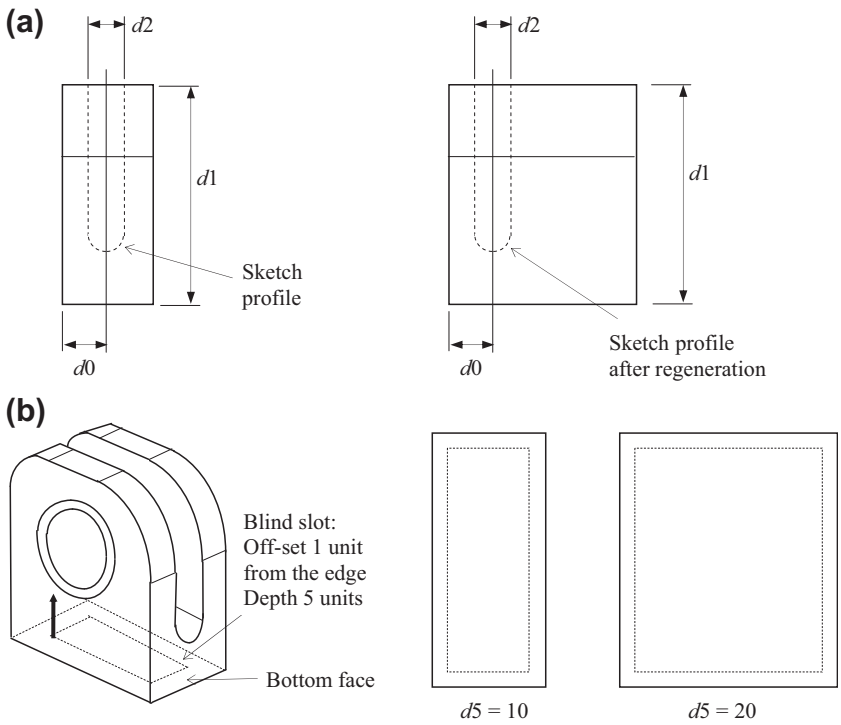


FIGURE 3.22

Feature rebuilds. (a) Side cut. (b) Bottom cut.

illustrated in Figure 3.22(a). The intersecting curves of the base block and the cut features are computed.

4. Fourth feature: chamfer
   This feature is not affected because the placement edges (the circle on the front face) and the size of the chamfer are unchanged.
5. Fifth feature: cut on the bottom face. Check the sketch profile. Is the sketch profile changed? Yes, because the rectangular profile is defined with an offset from the exterior rectangle, as illustrated in Figure 3.22(b). The system of equations will be resolved for the four characteristic points (corner points of the rectangle) due to the change of *d*5. The cut feature is regenerated following the feature attributed—that is, blind with depth 5 and one side with the same cut direction. The intersecting curves are computed.

If we change the width of the base block (*d*5) from 10 to 0.3, which feature(s) will fail to rebuild? Both the side cut and bottom cut features will not be generated because there is not enough room for the sketch profiles to be generated. However, CAD usually stops at the first unsuccessful feature; therefore, in this case, the rebuild error on the side cut will be reported.

3.3.6 DIRECT MODELING

As can be seen from the discussion above, the parametric modeling approach requires the designer to anticipate design changes and accordingly define features, add relations to sketch entities, and add parameter relations between features. As a result, the solid model is created in such a way that a design modification (e.g., change in a dimension value) triggers rebuild in solid features in a prescribed manner. Feature-based parametric modeling is a structured modeling process, in which feature creation sequence or history tree masters the model rebuild process and design intent is captured implicitly through sketch relations, parent–child relationships, and parametric relations between dimensions.

Although the feature-based parametric modeling is indispensable in support of product design in the e-Design paradigm, capturing design intents in complex solid models is not always straightforward, to say the least. It requires the designer's effort, considerable planning, and careful implementation in achieving such parametric solid models. In general, parametric CAD tools lack ease of use, speed, and modeling flexibility. It requires a relatively steep learning curve and modeling effort upfront for the designer, and the solid models created suffer from model interoperability issues; that is, a CAD model created in software A cannot be understood or imported to software B with features and dimensions due to the nature of the "history-based" model.

The newly developed direct modeling approach provides a geometric-based modeling strategy that gives designers the power to quickly define and edit geometry by simply clicking on the model geometry and moving it with a mouse. Designers can focus on creating geometry rather than building features, adding constraints and design intent into their models and therefore speeding up design, saving time and development costs, and increasing productivity. The direct modeling paradigm is especially suited to the needs of designers working with legacy and heterogeneous CAD data. The direct modeling eliminates the need to access feature-level information to implement design changes. Designers can easily edit, modify, and repurpose solid models from any CAD sources.

Both Pro/ENGINEER (Creo™ 2.0 and higher) and SolidWorks (2012 and newer) are equipped with direct modeling (also called direct model editing) capabilities, which is built on top of existing feature-based parametric modeling technique. With the added direct modeling capability, designers are able to copy, move, split, replace, offset, push, and drag geometry to create the result as desired, instead of clicking on a dimension, entering a different value, and asking for model rebuild. In addition, with direct modeling capability, CAD automatically imports nonnative, imported model geometry without a model tree. The imported geometric model can be modified through direct geometry manipulation.

In general, parametric modeling is a history-based modeling method that enables design automation and creates product platforms for a product family, which are suitable for a product design strategy that is aimed to be family-based or platform-driven. On the other hand, direct modeling is a geometry-centered and history-free approach that supports quick and easy 3D solid model construction, allows design change through direct manipulation of geometric models, and supports direct geometry-editing from any CAD sources. There are pros and cons to these two methods. They are not exclusive but in general complement each other. More details can be found in Projects S1 and P1.

3.3.7 GEOMETRIC MODELING KERNELS

No matter what kind of modeling method is offered by a CAD system, the core of any CAD software is its geometric modeling kernel. The kernel is key to support underline computing and modeling capabilities of solid objects, as well as output or export solid models, including 2D drawings, from 3D

geometry. All commercially available solid modeling systems today are built on top of a geometric modeling kernel (also referred to as a modeling engine or geometry library). This is the library of core mathematical functions that defines and stores 3D solid objects in response to users' commands. The kernel library processes commands input through the application's user interface, stores the results, and submits the output to the graphics package for display, as illustrated in Figure 3.23.

Following Figure 3.23, it is commonly understood that there are two layers of information created when designers work with CAD. The top layer records user interaction with the CAD through either feature-based or direct modeling capabilities in the form of geometric features, including sketches, attributes, parameters, and equations, and feature construction sequence or history tree. On the bottom layer is the resulting geometric entities or objects. A history-based CAD system is basically recording every function it sends to the kernel into the history tree. For example, a sketch with an extrusion distance creates an extruded feature. Constraints control the size and position of the new feature. Then, a Boolean function is added to specify whether the feature is added or removed from the parent geometry. The Boolean function with the feature and related parameters are passed to the kernel and the resulting geometry is processed and revealed. This kernel function is processed every time this feature is rebuilt. The kernel function along with its required parameters is very specific to the kernel as discussed above. It is highly unlikely that another kernel will understand this very specific function, and even if it did the geometrical results could be very different. This is one of the major issues to address in solid model interoperability among CAD systems.

Some geometric modeling kernels such as ACIS (Spatial Inc.), Parasolid (Unigraphics Solutions, Inc.), and SMLib (Solid Modeling Solutions) are licensed by their respective developers for use in many different CAD systems. Others, such as thinkernel (think3), Granite One (Pro/ENGINEER), and UPG2 (Varimetrix Corporation), are proprietary kernels developed exclusively for a specific CAD system.

Apart from the underlying functionality supported, both licensed and proprietary kernels offer distinct advantages. CAD systems that license the same kernel can directly exchange model files that the kernel generates. For example, you can load SAT (ACIS) files directly into a CAD system that uses the ACIS kernel. On the other hand, developers who use different kernels in their CAD systems must write specific translators to read and write model files for import and export. Exporting and importing parametric solid models is not straightforward. More about CAD model translation is discussed in Chapter 6.

The geometric modeling kernels adopted by major commercial CAD systems are summarized in Table 3.1. In the following, we briefly introduce the two kernels that are widely employed in CAD: ACIS and Parasolid.

ACIS is an object-oriented C++ geometry library that integrates wireframe, surface, and solid modeling with both manifold and nonmanifold topology. It gives application developers a rich set of geometric operations for constructing and manipulating complex models. These include extruding,

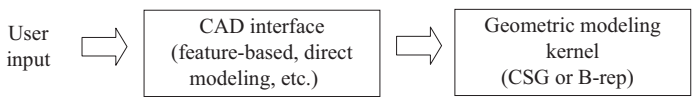


FIGURE 3.23

Relationship of a CAD interface to geometric modeling kernels.

Table 3.1 Geometric Modeling Kernels and CAD Systems

CAD Systems	Software Developer	Geometric Modeling Kernel
AutoCAD 2000	AutoDesk	ACIS
Pro/ENGINEER	PTC	Granite one
I-DEAS	SDRC	Geomod
Unigraphics	EDS	Parasolid
SolidWorks	Dassault systems	ACIS
CATIA	Dassault systems	CGM (convergence geometric modeler)
NX	Siemens	Parasolid

sweeping, lofting, skinning, offsetting, slicing, stitching, sectioning, fitting, and interpolating surfaces. ACIS also offers a complete set of Boolean operations, and length, area, and mass property inquiry functions. The ACIS kernel outputs a SAT file format that any ACIS-enabled application can read directly.

Parasolid from Unigraphics Solutions is an exact B-rep modeler that supports solid modeling and integrated free-form surface and sheet modeling. Parasolid's Extreme Modeling is a set of tightly integrated, proprietary technologies that enable modeling of complex geometry. Parasolid comprises more than 600 object-oriented functions for applications running on Windows, UNIX, and LINUX.

3.4 SOLID MODEL BUILD PLAN

One of the objectives in creating solid models for product design using the e-Design paradigm is to capture design intents so that design changes can be made by simply modifying a few dimension values and rebuilding the solid models. To achieve the objective, it is important for the designer to plan ahead and spell out detailed steps in terms of the type of features, their sketch profiles, relations and dimensions, parent–child relationships, and the feature creation sequences, including the construction of datum features. This is especially true when we use a CAD system with a feature-based parametric modeling method, which is still the mainstream technology implemented in major CAD systems.

Why do we need to spend time in developing such a part build plan? First, we always want to complete our work in the minimum time with the best result. If we plan ahead and think through the best possible way to create a design in using a CAD system, we often foresee possible pitfalls and are able to take precautions, which save us time in the end by not throwing out the problematic and incomplete models and starting over. So, please think before you do it! Planning ahead saves you time and makes you a better CAD user.

Let us take a look at the two examples in Figure 3.24. How should we construct the hose support and bracket shown in Figures 3.24(a) and (b), especially when a design change is anticipated for the bracket, as illustrated in Figures 3.25(a) and (b)? After all, we are not just creating a part, we are creating a quality part that is clean, organized, and well thought of.

What do we mean by a quality part? A quality part must be accurate in revealing geometric features in support of product design and manufacturing. Its feature construction sequence must be logical so

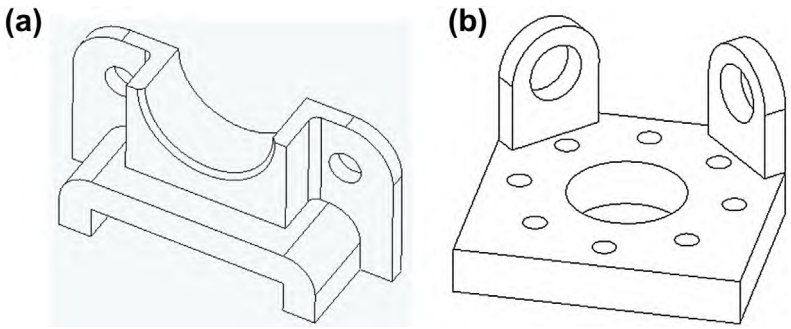


FIGURE 3.24

Sample parts for illustrating part build plan: (a) hose support and (b) bracket.

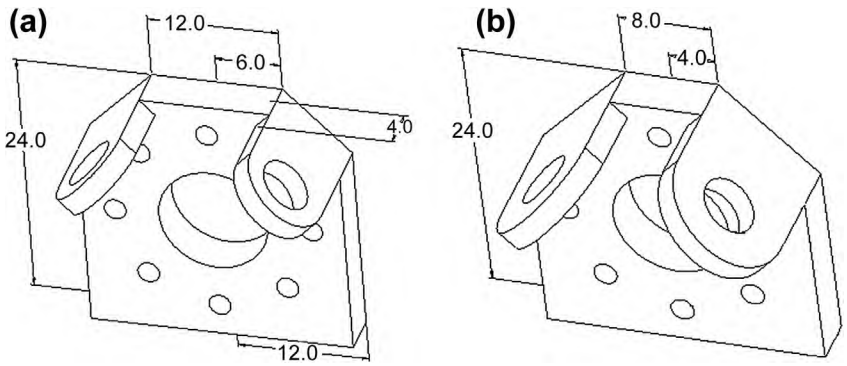


FIGURE 3.25

Design intent for the bracket shown in Figure 3.24(b). (a) Dimension design variable. (b) Dimension changed from 12 to 8.

that other team members can understand the solid model. The part should have the minimum number of features. However, it does not imply that you must create complex sketches. It is a tradeoff in minimizing the number of geometric features and complexity level of individual sketch profiles. Moreover, a quality part must have the minimum number of dimensions, implying as many sketch relations as possible. Most important, a quality model must be correctly parameterized and capture design intents. More about design parameterization is discussed in Chapter 5.

What should be included in the part construction plan? At least three things must be included:

1. Features and feature creation sequences (also include construction or datum features)
2. Sketch of each feature, including sketch plane, geometric entities, dimensions, and relations
3. Equations between dimensions as needed.

After a plan is jotted down on paper, it is a good idea to review it and try to optimize it before implementing it in CAD. The plan does not have to be fancy; it does have to facilitate your work. Note that a part construction plan is not unique. There is no "best plan".

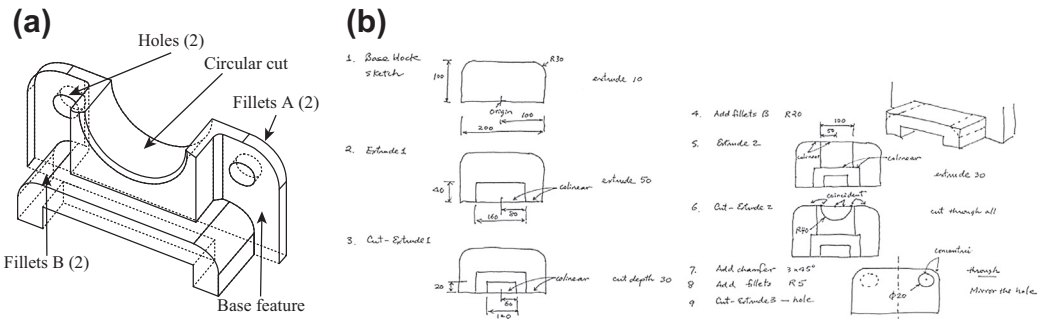


FIGURE 3.26

Build plan for the hose support shown in Figure 3.24(a). (a) Major features. (b) Hand-sketched sample build plan.

Let us take a look at the example part: hose support shown in Figure 3.24(a). Usually the first question to come to mind is how many solid features need to be created and which one the base feature (first solid feature) should be. For this part, it seems to be logical to create the back plate as the base feature (see Figure 3.26(a)). Is it a good idea to create the semicircle in the sketch of the base block? Is it a good idea to create fillets A in the sketch? How about the holes? Is it a good idea to take advantage of part symmetry by focusing on only half of the part and then mirror the first half for the remaining half? Again, there is no "correct" answer to these questions. The key principle is creating a quality part with a minimum of effort. A hand-sketched sample build plan for the hose support example is given in Figure 3.26(b) for your reference.

Now, let us discuss the bracket example. The detailed dimensions are provided in Figures 3.27(a) and (b).

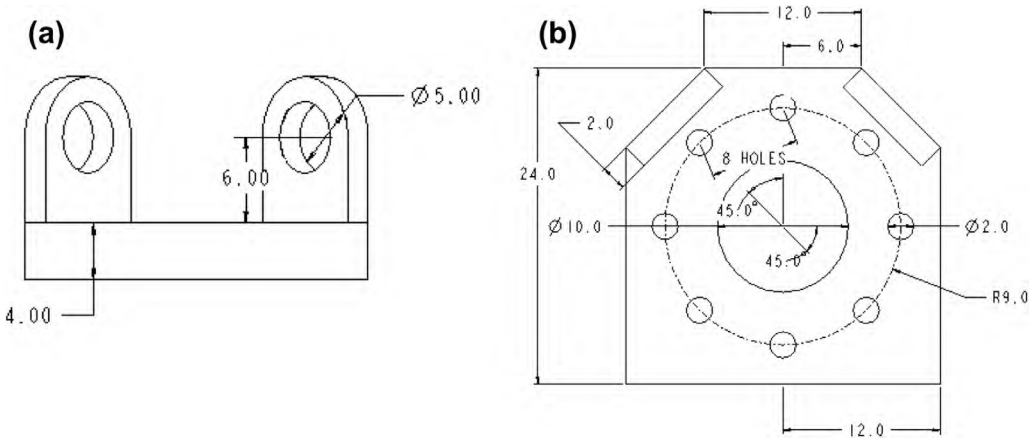


FIGURE 3.27

Sample part: bracket with dimensions. (a) Front view. (b) Top view.

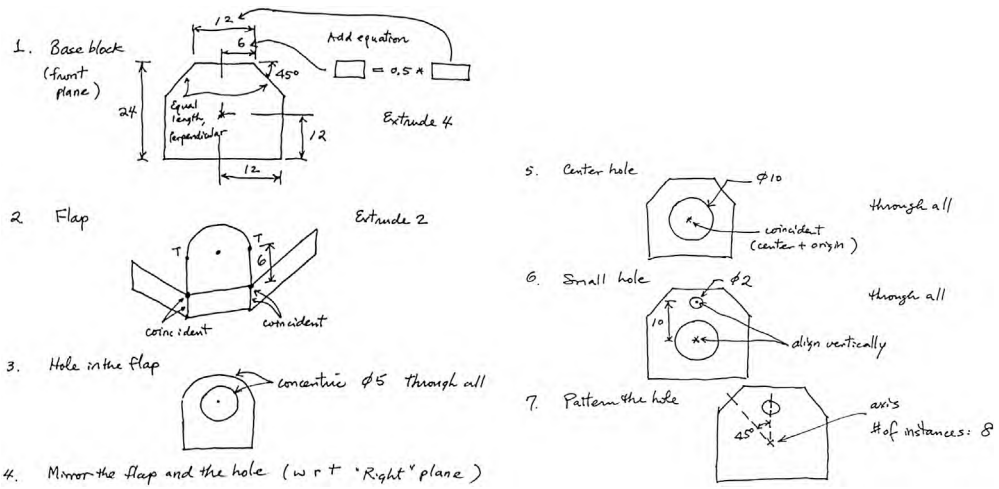


FIGURE 3.28

Hand-sketched build plan for sample part: bracket.

Note that the design variable is the length dimension of the top edge (current value: 12.0, shown in Figure 3.27(b)). When the design variable is changed, we expect to see the changes shown in Figure 3.25(b); that is:

1. The 45° edges of the base block remained 45°, but with their lengths changed in accordance with new design variable value.
2. The back faces of the flaps align with the 45° edge of the base block.
3. The hole in the flaps stays in the middle with size unchanged.

How do we create such a part with the requirements described above? A sample build plan is shown in Figure 3.28 for your reference. To a less experienced CAD user, for a part like the bracket shown in Figure 3.24(b), if you do not think ahead, you may end up throwing away several "wrong" models before actually creating one that works. So please think before you actually construct a solid model in front of a computer.

3.5 COMMERCIAL CAD SYSTEMS

Since the early 1960s when the first wireframe computer graphics was invented at MIT's Lincoln Laboratory, CAD has advanced significantly and has become the de facto design tool for the industry around the world. The first commercial applications of CAD were in large companies of automotive and aerospace industries, as well as in electronics. Only large corporations could afford the computers capable of performing the calculations. Notable company projects were at GM with DAC-1 (Design Augmented by Computer) in 1964 and at Renault–UNISURF 1971 car body design and tooling.

As computers became more affordable, the application areas have gradually expanded. The development of CAD software for personal desktop computers was the impetus for almost universal application in all areas of engineering. The most significant development appeared in the mid-1990s,

in which major CAD tools were made available in PCs that allowed end users in mid- and small-size companies to be able to bring design from the drawing board into digital form.

3.5.1 GENERAL PURPOSE CODES

Several general purpose commercial CAD systems available today were developed decades ago. These systems include the solid modeling packages Romulus (ShapeData) and Uni-Solid (Unigraphics), and the release of the surface modeler CATIA (Dassault Systemes) in 1981. Autodesk was founded in 1982, which led to the 2D system AutoCAD. Integrated Design and Engineering Analysis Software (I-DEAS) was produced by Structural Dynamics Research Corporation in 1982, and used primarily in the automotive industry, most notably by Ford Motor Company and General Motors. The next milestone was the release of Pro/ENGINEER in 1988, which heralded greater usage of feature-based modeling methods and parametric linking of the parameters of features. Also of importance to the development of CAD was the development of the B-rep solid modeling kernels Parasolid (ShapeData) and ACIS (Spatial Technology Inc.) at the end of the 1980s and beginning of the 1990s. This led to the release of mid-range packages such as SolidWorks in 1995, Solid Edge (then Intergraph) in 1996, and Autodesk Inventor in 1999. Several major mergers occurred throughout the years. SDRC was bought in 2001 by Electronic Data Systems, which had also acquired UGS Co. (maker of Unigraphics). EDS merged these two products into NX. UGS was purchased by Siemens AG in 2007 and was renamed Siemens PLM Software.

All major CAD systems offer not only solid modeling capabilities, but also CAE and CAM. Some CAD systems are equipped with in-house CAE/CAM, such as CATIA and Pro/ENGINEER. Some CAD partners with third-party software developers and fully integrates the third-party codes to the system; for example, CAMWorks integrated with SolidWorks. Although all major CAD systems offer excellent solid modeling and CAE/CAM, they serve different industrial sectors with slightly different focuses. CATIA is widely used by aerospace and automotive industry because of its superior surface modeling capabilities. AutoDesk is popular in small and mid-size companies due to its excellent capability in 2D drafting and its availability on PC in early years. Pro/ENGINEER serves heavy equipment industry, such as Caterpillar, due to its pioneering parametric modeling technology and strong CAE in the 1990s. SolidWorks became popular in almost all industrial sectors, as well as academia, because the software is intuitive and easy to use.

Several review articles on CAD software tools, such as those offered by 10 Top Ten Reviews (cad-software-review.toptenreviews.com), Cadalyst (www.cadalyst.com/listing/9/3d-modeling), and Wikipedia (http://en.wikipedia.org/wiki/List_of_computer-aided_design_editors, and http://en.wikipedia.org/wiki/Comparison_of_3D_computer_graphics_software), provide in-depth reviews and comparisons among major commercial systems. Readers are strongly encouraged to take a look at these articles for a better understanding of commercial CAD software.

3.5.2 SPECIAL CODES

Besides general-purpose CAD software tools, there are at least two special codes worth mentioning. They are Rhinoceros (www.rhino3d.com) and SpaceClaim Engineer (www.spaceclaim.com/en/default.aspx).

Rhinoceros (Rhino) is a stand-alone, commercial nonuniform rational B-spline (NURB)-based 3D modeling software, commonly used for industrial design, architecture, marine design, jewelry design,

automotive design, as well as the multimedia and graphic design industries. Rhino specializes in free-form NURB modeling. Rhino is gaining popularity due to its diversity, multidisciplinary functions, low learning curve, relatively low cost, and its ability to import and export many file formats, which allows Rhino to act as a "converter" tool between programs in a design workflow.

SpaceClaim Engineer is a 3D direct modeler. It enables engineers to easily create concepts and prepare 3D designs for prototyping, analysis, and manufacturing without becoming experts in traditional feature-based CAD systems. SpaceClaim helps engineers interact with CAD geometry in new ways. Without becoming a CAD expert, users can edit models, conceptualize on-the-fly, and communicate quickly and easily with prototyping and manufacturing. Direct modeling changes the way designers think about working with 3D solid models by letting them focus on what they are designing. Intuitive tools such as Pull and Move let users directly select portions of the model and move them where users want. The Combine tool slices and divides parts into pieces and lets users merge in portions from other designs. The Fill tool cleans up small features and fills holes. Together, these direct modeling tools let designers get the job done without resorting to traditional CAD.

3.6 SUMMARY

There is no doubt that CAD offers a better visualization of the design, easier creation of drawings once the model is completed, and better integration with CAE and CAM for product development. We discussed in this chapter the fundamentals in solid modeling, including CSG and B-rep, the two most commonly employed methods for underline solid modeling in CAD. We introduced the mainstream solid modeling technique—feature-based parametric solid modeling—that is employed in major CAD systems. We discussed key concept and theories, including variational and parametric modeling techniques, the parent–child relationship, and feature construction sequence or history tree. We walked through the steps of model rebuild in CAD using a simple example. We also briefly introduced geometric modeling kernels and newly developed direct modeling method. In addition, we offered a brief overview of commercial CAD systems. We hope you have gained adequate knowledge of CAD and solid modeling techniques and understand the behind-the-scenes operations that CAD carries out when you interact with it.

It is important to point out that although CAD becomes essential for product design, especially using the e-Design paradigm, it has a few issues. First, CAD can be slow for conceptual design. In the early stages, we tend to think faster than anybody could model in 3D. The direct modeling method may offer good alternatives to this issue. Also, CAD may require a lot of computing power to handle complex parts and assemblies. Display and rendering such models can be slow and model rebuild due to design changes can be too sophisticated to handle. Finally, model interchange between 3D parametric CAD systems is still an open issue. More about CAD interoperability is discussed in Chapter 6. We are now ready to move on to the next chapter to discuss the theory and methods employed by CAD for assembly. Our goal again is to understand the behind-the-scenes operations in CAD when we use it for creating assemblies.

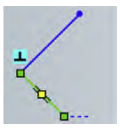
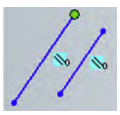
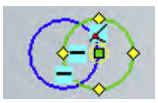
APPENDIX 3A: SKETCH RELATIONS

Sketch relations play an important role in solid modeling. In this appendix, we offer tables that illustrate the commonly seen relations (also called sketch constraints) in SolidWorks and Pro/ENGINEER in Table 3A.1. Examples of such relations in SolidWorks are provided in Table 3A.2.

Table 3A.1 Commonly Seen Sketch Relations (or Constraints) in *Pro/ENGINEER* and *SolidWorks*

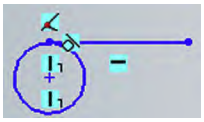
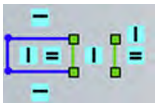
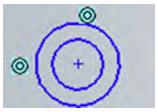
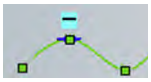
Pro/E	*SolidWorks*	Name	Entities	Descriptions
H or V		Horizontal or vertical	One or more lines or two or more points	The lines become horizontal or vertical
		Alignment	Two or more vertices	The items are aligned vertically or horizontally
⊥		Perpendicular	Two lines	The two items are perpendicular to each other
//		Parallel	Two or more lines	The items remain parallel
T		Tangent	An arc, ellipse, or spline, and a line or arc	The two items remain tangent
⊕		Concentric	Two or more arcs, or a point and an arc	The circles and/or arcs share the same centerpoint
		Coincident	A point and a line, arc, or ellipse	The point lies on the line, arc, or ellipse
*L*1 *L*1 *R*1 *R*1		Equal	Two or more lines, or two or more arcs	The line lengths or radii remain equal
→ ←	N/A	Symmetric	A centerline and two points, lines, arcs, or ellipses	The items remain equidistant from the centerline, on a line perpendicular to the centerline

Table 3A.2 Examples of Sketch Relations Seen in *SolidWorks*

Relations	Icons	Notes
Horizontal		Horizontal line sketched
Perpendicular		Second line was sketched perpendicular to the first. Sketch tool is active, so midpoint sketch snap is displayed on line
Parallel		Two lines sketched with parallel relation
Horizontal and tangent		Tangent arc added to horizontal line
Horizontal and coincident		Second circle. Sketch tool is active, so quadrant sketch snaps display on the second arc

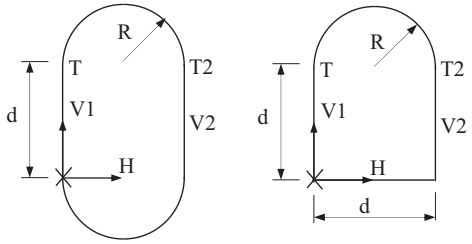
Continued

Table 3A.2 Examples of Sketch Relations Seen in *SolidWorks*—cont'd

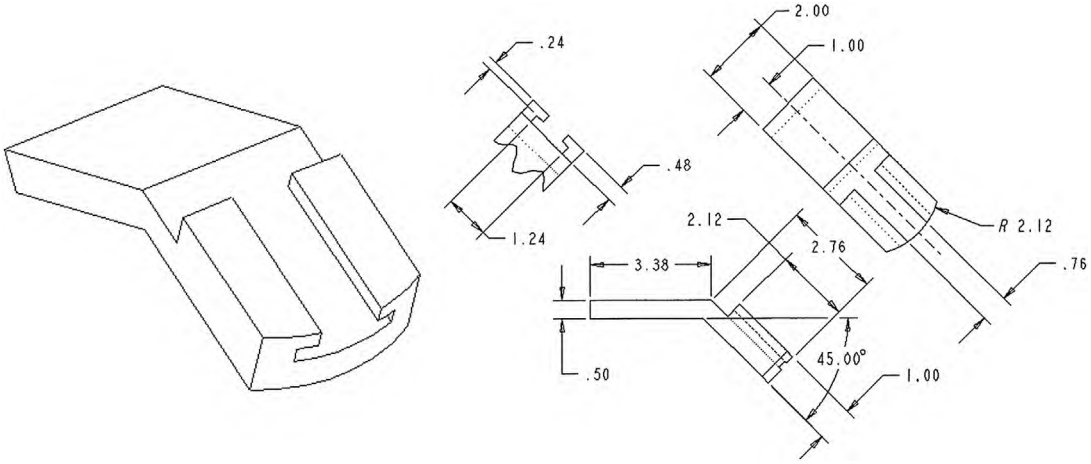
Relations	Icons	Notes
Vertical, horizontal, intersection, and tangent		Circle sketched with center inferred to sketch origin (vertical) Horizontal line intersects circle quadrant Tangent relation added
Horizontal, vertical, and equal		Horizontal and vertical relations inferred Equal relation added
Concentric		Concentric relation added
Horizontal		Horizontal relation added to spline handles

QUESTIONS AND EXERCISES

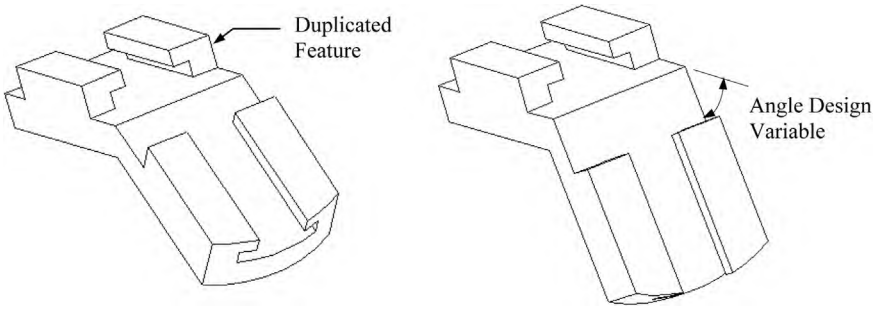
3.1. Are the following sketches *fully defined*? Why or why not? Please answer the question by formulating equations similar to those discussed in Section 3.3.



3.2. Create a solid model shown below. Make sure your model has identical dimensions as shown.



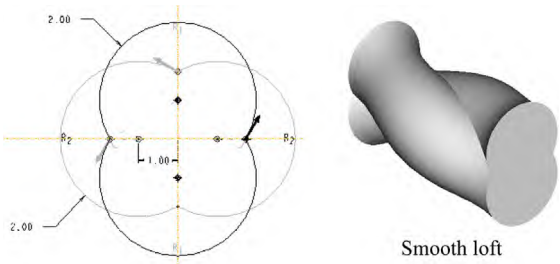
a. Duplicate the extrusion feature to the top flat surface of the part as shown below (left).



b. Define necessary relations so that when the angle design variable changes, the copied feature stays on the surface (above, right).

c. Submit four views, front, top, side, and isometric of the final part (including hidden lines; angle design variable = 45°). Submit a screen capture that shows an equation employed to capture the angle design variable.

3.3. Create a solid model with a smooth loft feature using the exact dimensions shown in the sketch. Note that there are three sections in the loft, and the distances between them are 5 units for each pair. Note that you will have to state how many guide curves if any are defined to create such a loft.



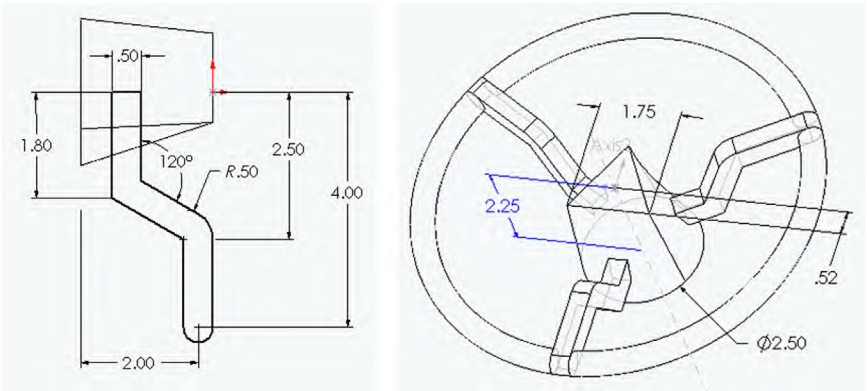
a. Define relation(s) so that modifying a single dimension can change the diameter of all arcs. Note that the relation(s) must be implemented in the solid model and you must show screen captures of the solid model before and after changes, with a single dimension change.

b. What is the range of the diameter dimension that you can change that results in valid solid models?

3.4. Create a steering wheel solid model using the exact dimensions shown in the figures below.

Note that the width of the spoke is 0.4; the triangle on top of the hub is an equilateral triangle. The bottom sketch profile of the hub is a circle with diameter 2.5. The distance between the bottom edge of the triangle and the origin is 0.52, and the height of the hub is 2.25 as shown below.

a. What is the outer diameter of the wheel? _____



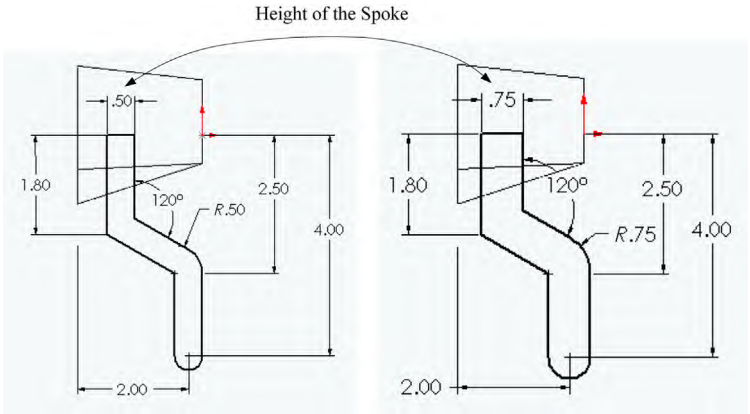
b. Define relations and equation(s) so that when the height dimension of the spoke is changed— for example, from 0.5 to 0.75—the sketch profile of the spoke can be regenerated as shown.

Relations: Please show relations on the sketch with brief explanation.

Equation(s): Please list all equations and point the dimensions involved in the sketch or part.

What is the range of the outer diameter of the wheel that you can change that results in valid solid models?

Minimum = ¼ _____, Maximum: _____



c. Please submit four views, front, top, side, and isometric of the final part (including hidden lines).

REFERENCES

Bozdoc, M., 2003. The History of CAD. www.mbinfo.mbdesign.net/CAD-History.htm.

Lee, W., 1999. Principles of CAD/CAM/CAE Systems. Addison Wesley Longman, Inc.

Planchard, D.C., Planchard, M.P., 2013. Official Certified SolidWorks Professional (CSWP) Certification Guide with Video Instruction. SDC Publications.

Pratt, M.J., Wilson, P.R., 1985. Requirements for Support of Form Features in a Solid Modeling System. CAM-I, R-85-ASPP-01.

Toogood, R., Zecher, J., 2012. Creo Parametric 1.0 Tutorial and MultiMedia DVD. SDC Publications.

CHAPTER

ASSEMBLY MODELING

4

CHAPTER OUTLINE

4.1 Introduction 170
4.2 Assembly Modeling in CAD 172
4.2.1 Mating Constraints 173
4.2.2 Kinematic Joints 178
4.3 Assembly Modeling Technique 184
4.3.1 Transformation Matrix 185
4.3.1.1 Coincident 186
4.3.1.2 Concentric 186
4.3.1.3 Computation of the Transformation Matrix 188
4.3.2 Degree of Freedom Analysis 197
4.4 Kinematic Modeling Technique* 199
4.4.1 Mapping Mating Constraints to Kinematic Joints 200
4.4.2 D–H Representation 202
4.4.2.1 Open-Loop System 210
4.4.2.2 Closed-Loop System 214
4.4.3 Constructing the Joint Coordinate Systems 219
4.5 Case Study and Tutorial Example 225
4.5.1 Case Study: Virtual Reality 225
4.5.2 Tutorial Example: A Single-Piston Engine 227
4.6 Summary 228
Questions and Exercises 229
References 230

Assembly could mean very different things to different engineers. Mechanical engineers often consider mechanical assembly at the shop floor or assembly line, for which topics relevant to the physical assembly of a product—such as manual assembly vs automatic assembly, force and mass of parts, tools and equipment involved in assembly, tolerance analysis, and interference checking—are often the emphasis. Assembly process planning and assembly/disassembly are popular considerations for industrial engineers, who are often in charge of designing and running a product assembly line. Overall, it is essential for design engineers to acquire knowledge in these areas so that the practical aspects of product assembly can be incorporated into product design.

These topics, although important, will not be the focus of this chapter. For those who are interested in learning more about mechanical assembly or assembly process planning to enter this area for thesis

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00004-1
Copyright © 2015 Elsevier Inc. All rights reserved.

work, there are excellent references that provide in-depth reviews and discussions on various topics related to mechanical assembly, such as Dawari and Sen (2007) and Whitney (2004).

In this chapter, we focus on assembly modeling that addresses methods employed in computer-aided design (CAD) to represent assembly. Topics such as mating constraints, degrees of freedom (DOFs), and fully constrained vs underconstrained assemblies are included. In addition, we discuss methods that support design changes and kinematic analysis in CAD assembly, which are the two most common activities encountered in assembly modeling using CAD. We discuss both open-loop and closed-loop systems. Note that the methods discussed in this chapter are mainstream methods adopted in the CAD community; they do not necessarily represent a specific CAD system.

In addition to theoretical discussion, we include virtual reality (particularly the applications that support product design) as a case study to illustrate and demonstrate the application of CAD assembly for practical engineering designs. In addition, a single-piston engine assembly is employed as a tutorial example to illustrate the detailed steps in creating the assembly using both Pro/ENGINEER and SolidWorks. Detailed instructions for bringing up these models and steps for carrying out the assembly discussed in this chapter can be found in Projects P1 and S1 for Pro/ENGINEER and SolidWorks, respectively. Example models are available for download at the book's companion website http://booksite.elsevier.com/9780123820389.

This chapter was written with the assumption that readers are familiar with basic CAD operations in part modeling, especially using Pro/ENGINEER or SolidWorks. If this is not the case, we encourage you to go over examples presented in other books (e.g., Toogood and Zecher, 2012; Shih, 2013; Lombard, 2013, Reyes, 2013) before going over this chapter.

The overall objectives of this chapter are to (1) provide you with a general understanding of the methods that support assembly modeling in CAD, (2) familiarize you with the behind-the-scenes operations of CAD when a change is made or a part is dragged in an assembly, and (3) help you use Pro/ENGINEER or SolidWorks for creating basic assembly models (after going through the tutorial lessons).

4.1 INTRODUCTION

In the physical assembly of rigid parts, they are positioned (including location and orientation) relative to one another. The positioning of parts causes some of the low-level geometric entities, such as faces, edges, and vertices of the parts, to be in contact. The entities in contact between parts constrain the relative motion between them because a rigid part cannot deform or penetrate through other parts in the assembly.

The position of a part in space is uniquely defined by specifying its location and orientation with respect to some reference system. Three parameters are required to specify the location and another three parameters are required to specify the orientation. A rigid body in space has six degrees of freedom (DOFs) representing the allowable motions of the part. Assembly models are created by fixing the location and orientation of individual parts relative to one another through mating constraints, whereas kinematic models are created by specifying the allowed motions between the parts by defining kinematic joints.

CAD assembly has been commonly employed for product design. It is well known that the assembly design has a significant impact on many downstream activities, such as production process planning and control, tolerance analysis, and packaging. Assembly design involves the creation of assembly models that specify the relative location and orientation of components. In the design activity, component geometry is assembled together to create an assembly model. Mating constraints,

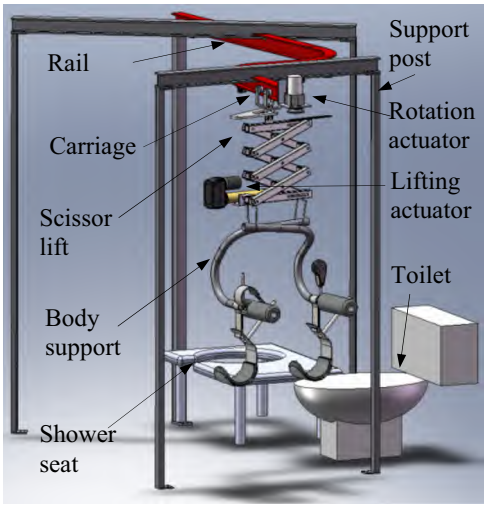


FIGURE 4.1

The bathroom transport device.

also called assembly mates (or placement constraints), are used to locate and orient components with respect to one another. With a CAD assembly, basic yet essential questions in design can be readily answered by the product assembled in CAD. For example, will the parts fit into the designated space as an assembly? Will the components of the assembly collide or interfere in operation? Will the assembly operate as intended?

A bathroom transport device shown in Figure 4.1, which was designed and manufactured by a team of undergraduate students as their capstone project, is used as an example to illustrate some of the points mentioned above. This device was created for the purpose of transporting a disabled woman from her wheelchair to the toilet and shower seat without human assistance. It also transports the person from the toilet or shower seat back to the wheelchair. The device is compact (to fit into a very small bathroom), durable, and tailored to help a person to overcome a physical disability. The design features a three-button remote control that will move the person to the toilet, shower, and back to the wheelchair; a scissor lift with a linear actuator that provides lift; a carriage on a rail system that carries the person to designated locations; and a body support that safely holds the person while the system transports her to designated locations. A second actuator mounted on top of the scissor lift provides a 90-degree rotation to the body support when the carriage is moved to the toilet so that the user will be properly oriented on top of the toilet. A motor and a cable system are employed to pull the carriage.

The design of the device was extremely challenging because it was made to accommodate a severely disabled person who can only use her right hand to operate the device. The person would pull her wheelchair to the entrance of the bathroom, right in front of the device, as shown in Figure 4.2. She would use her right hand to move the two leg supports under her thigh, place the two arm supports under her arms, and press a button on the remote control mounted on top of the right arm support. The button pressing triggers the actuator of the scissor lift to contract, creating a lift to move her out of the wheelchair. Then, a motor is activated to pull a cable that draws the carriage along the curve rail and

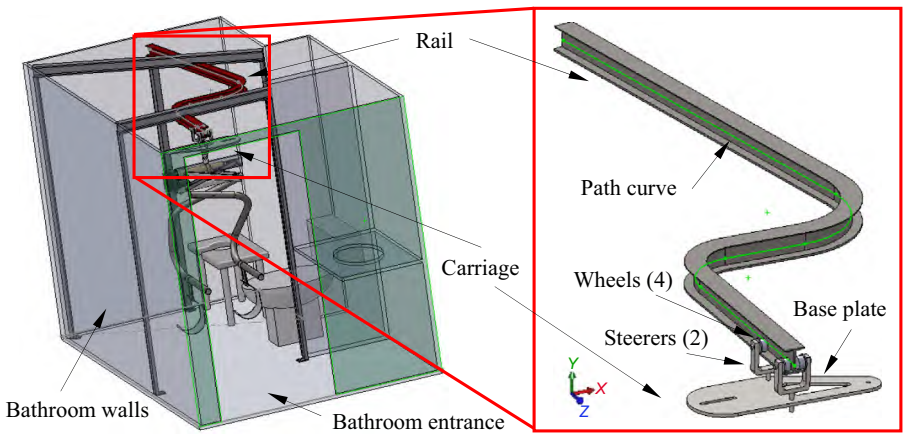


FIGURE 4.2

The rail and carriage subsystems.

transport her to the toilet or shower seat. Position sensors are mounted on top of the rail to detect the location of the carriage and activate motors or actuators for the desired motions.

While designing the device, path mates were employed to assemble the carriage to the rail, allowing the carriage to move along the rail. The rail and carriage are important features of the device. The rail is a curve I-beam, created by sweeping an I-cross section sketch along an open loop curve composed of three straight lines and two circular arcs, as shown in Figure 4.2. The carriage consists of a base plate, two steerers, and four wheels, as shown in Figure 4.2. The wheels are sitting on the top faces of the bottom flange of the rail. A cable connecting to a motor is pulling the steerers to move the carriage along the rail. A universal joint under the base plate connects the body support. This assembly model with motion animation helps verification of the design concept, facilitates communications within the design team, and supports demonstration of the device design to the sponsors and user.

In this chapter, we start with a short and brief introduction in Section 4.2 on the mating constraints and kinematic joints commonly offered by CAD systems. After becoming familiar with the constraints and joints, Section 4.3 discusses a method that supports the calculation of a transformation matrix that positions a mating part to the base part in space. This illustrates how CAD supports part assembly as designers bring individual components into an assembly and define mating constraints. In Section 4.4, we discuss a kinematic modeling technique, in which we introduce the conversion of a CAD assembly to a kinematic model, the mapping of mating constraints to kinematic joints, and the mathematical representation of a kinematic assembly in CAD. We include both open-loop and closed-loop systems. The chapter wraps up by introducing a case study that involves applications of virtual reality technology for product design. In addition, a tutorial example of a single-piston engine is provided.

4.2 ASSEMBLY MODELING IN CAD

In CAD, an assembly model is created by specifying the relative location and orientation of parts. In general, an assembly model is static, in which all parts are completely constrained (also called fully

constrained). On many occasions, the desired relative motion is required in product design to meet certain design requirements or verify functionalities. In these cases, an assembly is underconstrained, in which parts are allowed to move with respect to one another in order for the designer to explore or verify the kinematic characteristics of the assembly design.

A task common to both assembly modeling and kinematic analysis is the determination of part location and orientation satisfying certain constraints between these parts. There are two categories of geometric assembly relationships: geometry mating and joint mating. The former is usually static, whereas the latter allows relative motion and holds despite changes in the components' dimensions.

In general, geometry mating constrains geometric entities between mating parts. There are usually multiple pairs of entities constrained between the mating parts. On the other hand, joint mating constrains the relative motion between mating parts, instead of between geometric entities. As a result, there is one single joint between the two mating parts.

Some CAD systems, such as SolidWorks, support designers in creating an assembly model that is underconstrained so that the kinematic characteristics of the assembly can be explored by dragging individual parts. In other CAD systems, designers are required to complete the product assembly using mating constraints, and then convert the assembly model to a motion model by defining kinematic joints on top of the assembly in order to verify the kinematic characteristics of the assembly. This was the case, for example, in SolidWorks versions before 2008. In some CAD systems, such as Pro/ENGINEER, designers are given choices in either selecting mating constraints or kinematic joints or a mixed set to create the assembly model. If the assembly is intentionally created with an underconstrained status, components can be dragged and moved.

In Section 4.2.1, we introduce commonly employed mating constraints in CAD, especially SolidWorks and Pro/ENGINEER. Then, in Section 4.2.2, we provide readers with a list of standard and advanced mating constraints offered by SolidWorks for a more complete picture in terms of the kind of mating constraints you may expect to use. We use a slider-crank mechanism as an example to go over the assembly in both SolidWorks and Pro/ENGINEER. We also introduce kinematic joints and the associated DOFs they constrain. This section serves as a prelude to the theoretical discussion on the subject in Sections 4.3 and 4.4.

4.2.1 MATING CONSTRAINTS

There are six DOFs for each component in space: three translations and three rotations. In the geometry mating approach, users specify the relative positions of parts by interactively defining spatial relationships between the geometric elements of mating parts. The geometric elements used in geometry mating include points, planar faces, surfaces, and axes of cylinders and holes. Commonly employed mating constraints (or placement constraints in Pro/ENGINEER and assembly mates in SolidWorks) include coincident-mate, mate offset, coincident-aligned, concentric (or fit), angle, parallel, and align. These mating constraints are usually applied to the same type of geometric entities, such as a pair of planar faces for a coincident-mate, or different entities, such as a point on a curve for a path mate.

In CAD, the first part brought into the assembly is fixed to the default datum features with all six DOFs constrained. In Pro/ENGINEER, the first part can be assembled to the assembly datum features, such as datum planes or the datum coordinate system, using placement constraints (e.g., by aligning their respective coordinates). In SolidWorks, the first component is fixed by aligning the component coordinate system with the default coordinate system provided in the assembly (also called the world

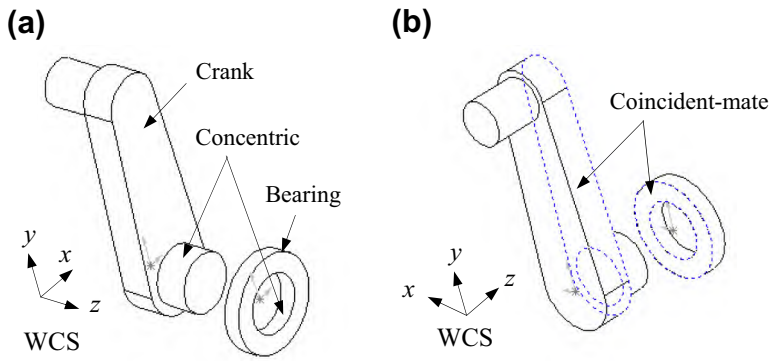


FIGURE 4.3

Mating constraints for the bearing and crank assembly. (a) Concentric, and (b) coincident-mate.

coordinate system or WCS). The first part serves as the base part for assembling other parts. When an existing part is brought into the assembly, there are an additional six DOFs associated with it for the designer to work with.

Most mating constraints restrict part motion between regular surfaces, such as flat surfaces and cylindrical surfaces. As a result, a mating part is allowed to translate or rotate along a fixed direction if it is underconstrained. For example, the lower shaft of the crank is to be inserted into the hole of the bearing, as shown in Figure 4.3. The bearing is fixed. The crank is assembled to the bearing using two mating constraints, concentric (called Mate: Concentric in SolidWorks and Insert in Pro/ENGINEER) and coincident-mate (called Mate: Coincident in SolidWorks and Mate or Align Surfaces in Pro/ENGINEER). The concentric mating constraint eliminates two translational DOFs and two rotational DOFs. The coincident-mate mating constraint eliminates one translational DOF and two rotational DOFs. As a result, only one DOF, Rz, remains, as summarized in Table 4.1. SolidWorks allows designers to move (rotate) the crank by simply dragging the part, according to the free DOF. The designer is able to check the kinematics of the product in the assembly mode. In Pro/ENGINEER, such a rotational DOF is allowed to be undefined; similarly in SolidWorks, components can be dragged to check the kinematic behavior of the assembly.

Note that in Figure 4.3(b), the coincident-mate mating constraint is more precisely called coincident with antialigned condition. In SolidWorks, you can set the alignment condition. The alignment conditions for a coincident mating constraint are either aligned, in which vectors normal to the selected faces point in the same direction; or antialigned, in which vectors normal to the selected faces point in

Table 4.1 Degrees of Freedom Eliminated by the Two Mating Constraints in Figure 4.3

	Tx	Ty	Tz	Rx	Ry	Rz
Mate: Concentric	×	×		×	×	
Mate: Coincident			×	×	×	

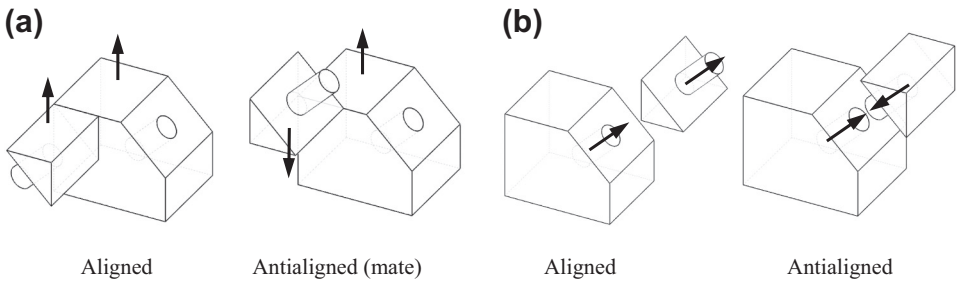


FIGURE 4.4

Aligned and antialigned conditions. (a) Between two flat faces, and (b) between two cylindrical surfaces.

opposite directions, as illustrated in Figure 4.4(a). For cylindrical surfaces, the axis vector is aligned or antialigned, as illustrated in Figure 4.4(b).

The most commonly used mating constraints in Pro/ENGINEER and SolidWorks are listed in Table 4.2. In addition, a complete list of standard mating constraints in SolidWorks with mate symbols is provided in Table 4.3. You may expect to use these mating constraints to create an assembly in SolidWorks and most modern CAD systems. In SolidWorks, mating constraints (standard) are imposed to surfaces, which are physically intuitive. In Pro/ENGINEER, in addition to surfaces, some mating constraints are applied to abstract geometric entities, such as point-on-surface and edge-on-surface constraints. In some cases, Pro/ENGINEER and SolidWorks will not accept the mate constraints as defined if they conflict with existing ones.

Table 4.2 Mating Constraints in Pro/ENGINEER and SolidWorks

Pro/ENGINEER	SolidWorks	Descriptions
Mate surfaces	Mate: Coincident, antialigned	Positions selected faces or planes so they coincide. Antialigned implies that the two faces or planes mate and the normal vectors of the two faces or planes point in the opposite directions, and aligned implies that the normal vectors of the two faces or planes point in the same directions
Align surfaces	Mate: Coincident, aligned	
Align axes or insert surfaces	Mate: Concentric	Places the selected cylindrical surfaces so that they share the common axis
Orient	Mate: Parallel	Places the selected items so they lie in the same direction and remain a constant distance apart from each other
Coordinate system	Default	Place the first part to the default coordinate system in assembly
Tangent	Mate: Tangent	Places the selected items in a tangent mate (at least one item must be a cylindrical surface)

Table 4.3 Standard Mates in SolidWorks

Standard Mates	Descriptions from SolidWorks Help
Coincident	Positions selected faces, edges, and planes (in combination with each other or combined with a single vertex) so they share the same infinite plane. Positions two vertices so they touch
Parallel	Places the selected items so they remain a constant distance apart from each other
Perpendicular	Places the selected items at a 90° angle to each other
Tangent	Places the selected items tangent to each other (at least one selection must be a cylindrical, conical, or spherical face)
Concentric	Places the selections so that they share the same center line
Lock	Maintains the location and orientation between two components
Distance	Places the selected items with the specified distance between them
Angle	Places the selected items at a specified angle to each other
Default	Places the first part to the default coordinate system in assembly

In this chapter, we adopt SolidWorks terminologies for mating constraints, except that we use *coincident-mate* instead of *coincident antialigned*.

In addition to standard mates, such as concentric and coincident, some CAD systems, such as SolidWorks, offer advanced mates, as listed in Table 4.4. Advanced mates provide additional ways to constrain or couple movements between parts. A coupler removes one additional degree of freedom from the kinematic model. For example, a linear coupler shown in Figure 4.5(a) removes one translational DOF by coupling the respective translational DOF between components 1 and 2. Also, path mate (one of the advanced mates in SolidWorks) allows a part to move along a curve slot, a groove, or fluting, varying its moving direction specified by the path curve. For example, in the rail and carriage assembly of the transport device shown in Figures 4.1 and 4.2, a vertex in the carriage is moving along the sweep curve (which can be either open- or closed-loop, composed of several

Table 4.4 Advanced Mates in SolidWorks

Advanced Mates	Descriptions
Symmetric	Forces two similar entities to be symmetric about a plane or planar face
Width	Centers a tab within the width of a groove
Path	Constrains a selected point on a component to a path
Linear/Linear coupler	Establishes a relationship between the translation of one component and the translation of another component
Limit	Allows components to move within a range of values for distance and angle mates

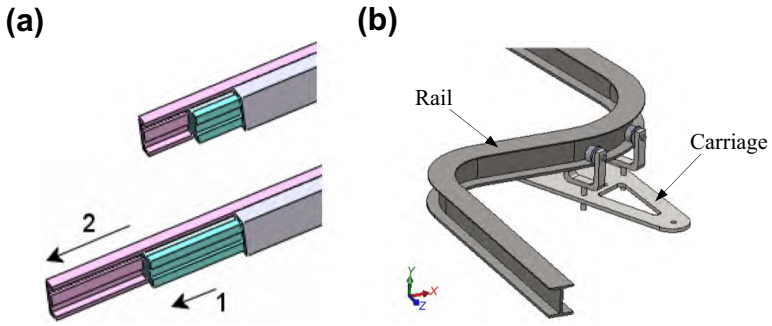


FIGURE 4.5

Examples of advanced mates in SolidWorks. (a) Linear coupler, and (b) path mate.

curves) of the rail, as shown in Figure 4.5(b). As a result, path mate allows the carriage to move along the curve groove of the rail, varying its moving direction specified by the path curve. In addition, the pitch, yaw, and roll of the moving part can be defined to resemble the physical conditions. Such a capability supports animation and kinematic analysis for a whole new set of applications that involves curvilinear motion.

Some CAD systems, such as SolidWorks and Pro/ENGINEER, also offer mechanical mates, such as cam follower, gear, hinge, rack and pinion, screw, and universal joint. These are essential for kinematic analysis of the product design. More about kinematic and dynamic analysis can be found in Chapter 8 Motion Analysis. Tutorial lessons can be found in Projects P2 and S2 for Pro/ENGINEER and SolidWorks, respectively. More tutorial lessons can also be found in Chang (2010).

Next, we use a slider-crank example shown in Figure 4.6 to illustrate the mating constraints employed for the assembly in SolidWorks. We will use the same example in Section 4.2.2 to illustrate the joint constraint approach for assembly as in, for example, Pro/ENGINEER. Note that model files of both examples are available for download at the book's companion website http://booksite.elsevier.com/9780123820389.

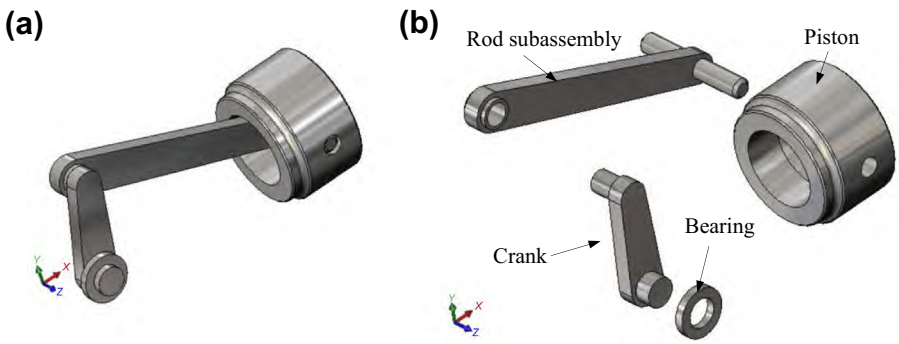


FIGURE 4.6

The slider-crank example. (a) Unexploded view, and (b) exploded view.

The slider-crank mechanism consists of five parts and one subassembly. They are bearing, crank, rod, pin, piston, and rod subassembly (consisting of rod and pin rigidly connected). An exploded view of the mechanism is shown in Figure 4.6(b). There are eight assembly mates, including five coincident and three concentric, defined in the assembly.

The first three mates (*Concentric1*, *Coincident1*, and *Coincident2*) assemble the crank to the fixed bearing, as shown in Figure 4.7(a). As a result, the crank is completely fixed. Note that the mate *Coincident2* orients the crank to the upright position, defining the configuration of the mechanism. Suppressing this mate will allow the crank to rotate with respect to the bearing.

The next two mates (*Concentric2* and *Coincident3*) assemble the rod to the crank, as shown in Figure 4.7(b). Unlike the crank, the rod is allowed to rotate with respect to the crank. The next two mates (*Concentric3* and *Coincident4*) assemble the piston to the pin, allowing the piston to rotate about the pin. The final mate (*Coincident5*) eliminates the rotation by aligning two planes, *Plane3* of the piston and the *Plane2* of the bearing, as shown in Figure 4.7(c).

At this point, the entire assembly is fully constrained. No relative motion between any components is allowed. If we suppress *Coincident2* defined between the right plane of crank and right plane of the bearing, the crank is allowed to rotate along the z-direction of the WCS. If you drag the crank (or any component), the entire assembly is moving, as illustrated in Figure 4.8.

4.2.2 KINEMATIC JOINTS

In some CAD systems, such as Pro/ENGINEER, designers are given an option in choosing either mating constraints between geometric entities (like those of Tables 4.2 and 4.3) or defining kinematic joints between components.

A kinematic joint is a connection between two components that imposes constraints on their relative movement. There are in general two kinds of joints—a lower pair and higher pair. Physically, a lower pair joint is used to describe the connection between a pair of rigid components when the relative motion is characterized by two common surfaces sliding over one another. Commonly employed lower pair joints include revolute (also called hinge or pin), prismatic (also called slider or translation), cylindrical, planar, spherical, and screw, as shown in Figure 4.9. On the other hand, higher pair joints describe joints with points or lines, such as a cam-follower joint.

A prismatic, slider, or translational joint (Figure 4.9(a)) requires that a line in the moving component (or mating part) remains colinear with a line in the fixed component (or base part), and a plane parallel to this line in the moving component maintains contact with a similar parallel plane in the fixed component. This restricts five DOFs on the relative movement of the links—two translational and three rotational—which therefore has one translational degree of freedom.

A revolute, hinge, or pin joint (Figure 4.9(b)) requires a line in the moving component to remain colinear with a line in the fixed component, and a plane perpendicular to this line in the moving component maintains contact with a similar perpendicular plane in the fixed component. This restricts five DOFs on the relative movement of the parts—three translational and two rotational—which therefore allows only one rotational degree of freedom.

A cylindrical joint (Figure 4.9(c)) requires that a line in the moving component remain colinear with a line in the fixed component. It is a combination of a revolute joint and a prismatic joint. This joint has two DOFs—one translational and one rotational.

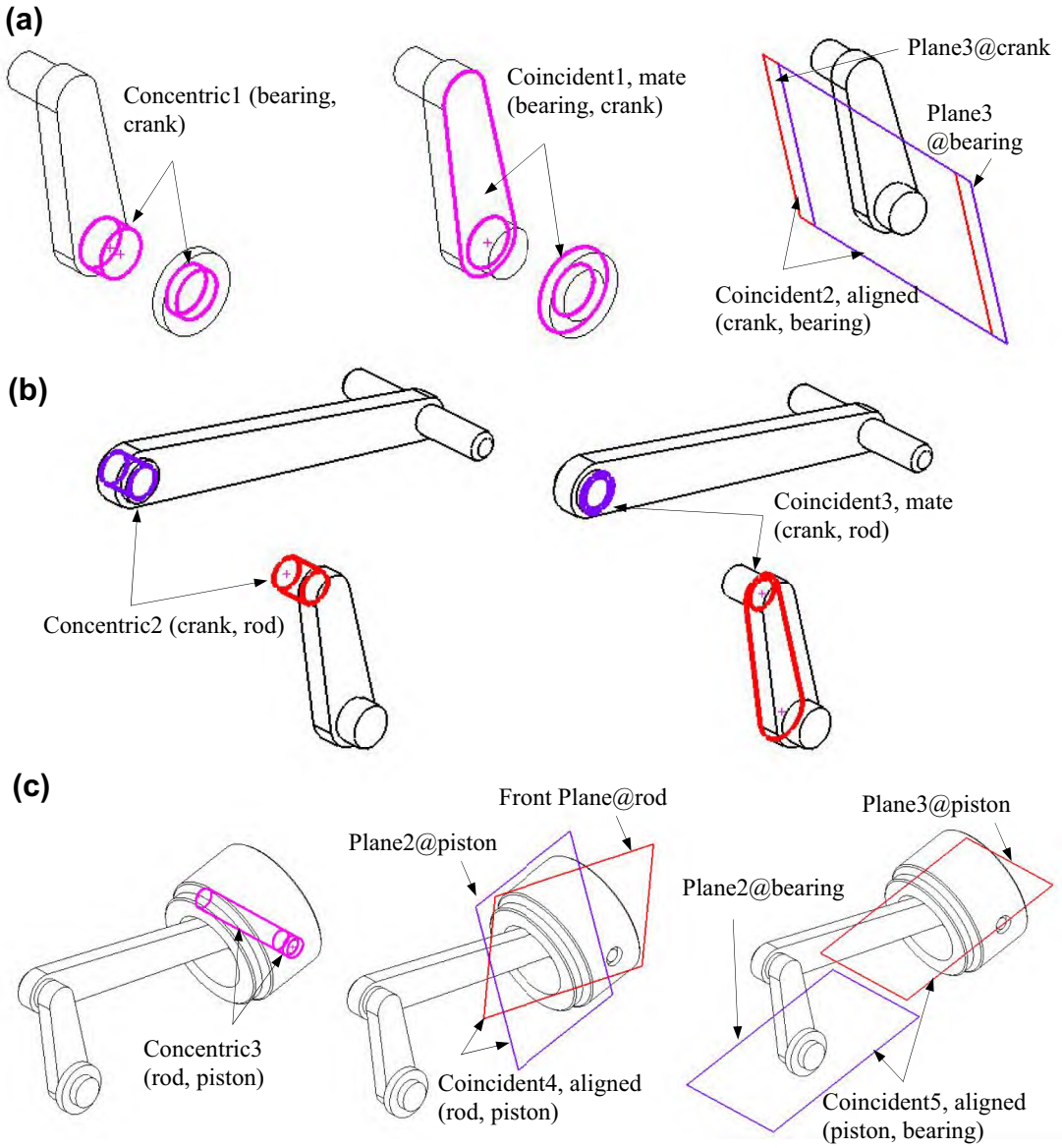


FIGURE 4.7

Assembly mating constraints defined for the slider-crank mechanism. (a) Mating constraints for crank (exploded view), (b) mating constraints for rod (exploded view), and (c) mating constraints for piston (unexploded view).

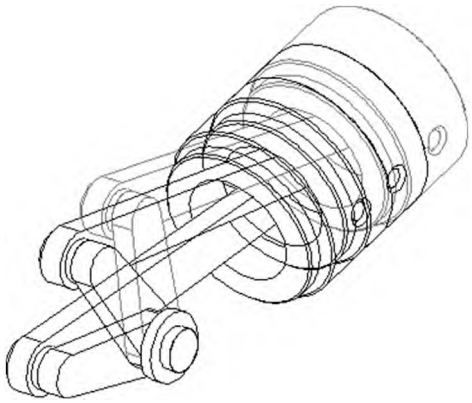


FIGURE 4.8

Drag the crank to explore the kinematic characteristics of the slider-crank mechanism in SolidWorks.

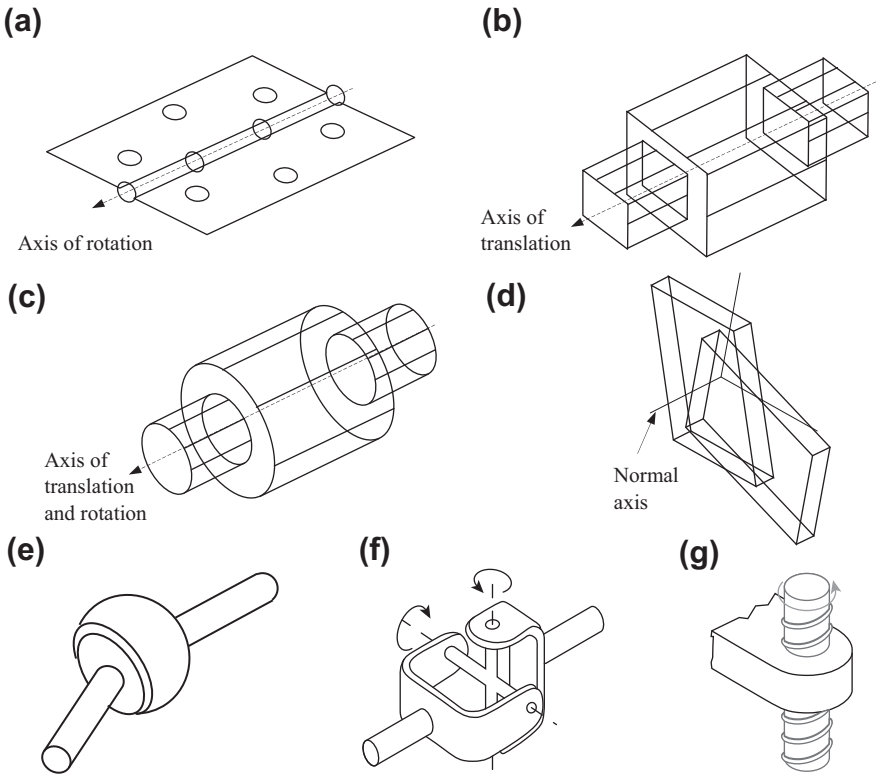


FIGURE 4.9

Lower pair kinematic joints. (a) Revolute, hinge, or pin, (b) prismatic, slider, or translational, (c) cylindrical, (d) planar, (e) spherical or ball, (f) universal, and (g) screw.

Table 4.5 Lower Pair Joints and the DOF Constrained

Joint Type	DOF Constrained			Remarks
	Translation	Rotation	Total	
Revolute	3	2	5	Rotates about an axis
Translational	2	3	5	Translates along an axis
Cylindrical	2	2	4	Translates along and rotates about an axis
Planar	2	1	3	Components connected by a planar joint move in a plane with respect to each other. Rotation is about an axis perpendicular to the plane.
Spherical	3	0	3	Rotates in any direction
Universal	3	1	4	Rotates about two axes
Screw	0.5	0.5	1	Coupled rotation and translation along one axis

A planar joint (Figure 4.9(d)) requires that a plane in the moving component maintain contact with a plane in the fixed component. This joint has three DOFs—two translational and one rotational.

A spherical joint, or ball joint (Figure 4.9(c)), requires that a point in the moving component maintain contact with a point in the fixed component. This joint has three DOFs—all rotational.

A universal joint (Figure 4.9(f)) allows the rotation of one component to be transferred to the rotation of another component. This joint is particularly useful to transfer rotational motion around corners or to transfer rotational motion between two connected shafts that are permitted to bend at the connecting point (such as the drive shaft in an automobile transmission system).

A screw joint (Figure 4.9(g)) requires cut threads in two components, so that there is a turning as well as sliding motion between them. This joint has one degree of freedom—coupled rotational and translational.

The DOFs that the lower joints constrained are summarized in Table 4.5.

Next, we use the same slider-crank example discussed in Section 4.2.1 to illustrate the kinematic joints employed for assembly in Pro/ENGINEER. Note that when using kinematic joints for assembly in Pro/ENGINEER, designers must use less physically intuitive entities, such as axis and points, to define joints.

Kinematically, the slider-crank example shown in Figure 4.6(a) is a four-bar linkage, as illustrated in Figure 4.10 schematically. They are commonly found in mechanical systems, such as internal combustion engines and oil-well drilling equipment. For the internal combustion engine, the mechanism is driven by a firing load that pushes the piston (slider), converting the reciprocal motion into rotational motion at the crank.

In the oil-well drilling equipment, a torque is applied at the crank. The rotational motion is converted to a reciprocal motion at the slider or piston that digs into the ground. Note that in any case the length of the crank must be smaller than that of the rod in order to allow the mechanism to operate. This is called Grashof's law (Erdman et al., 2001).

The slider-crank assembly shown in Figure 4.11(a) consists of four parts: crank (*crank.prt*), rod (*rod.prt*), pin (*pin.prt*), and piston (*piston.prt*), as shown in the exploded view in Figure 4.11(b). Instead of using a bearing part, we use the assembly datum features shown in Figure 4.11(c) as the ground. Datum points (such as *APNT0* in assembly and *PNT0* of crank shown in Figure 4.11(b)) and

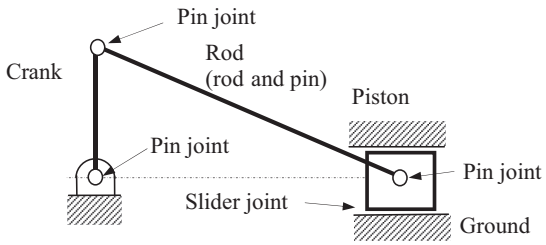


FIGURE 4.10

The schematic view of the kinematic model of the slider-crank mechanism.

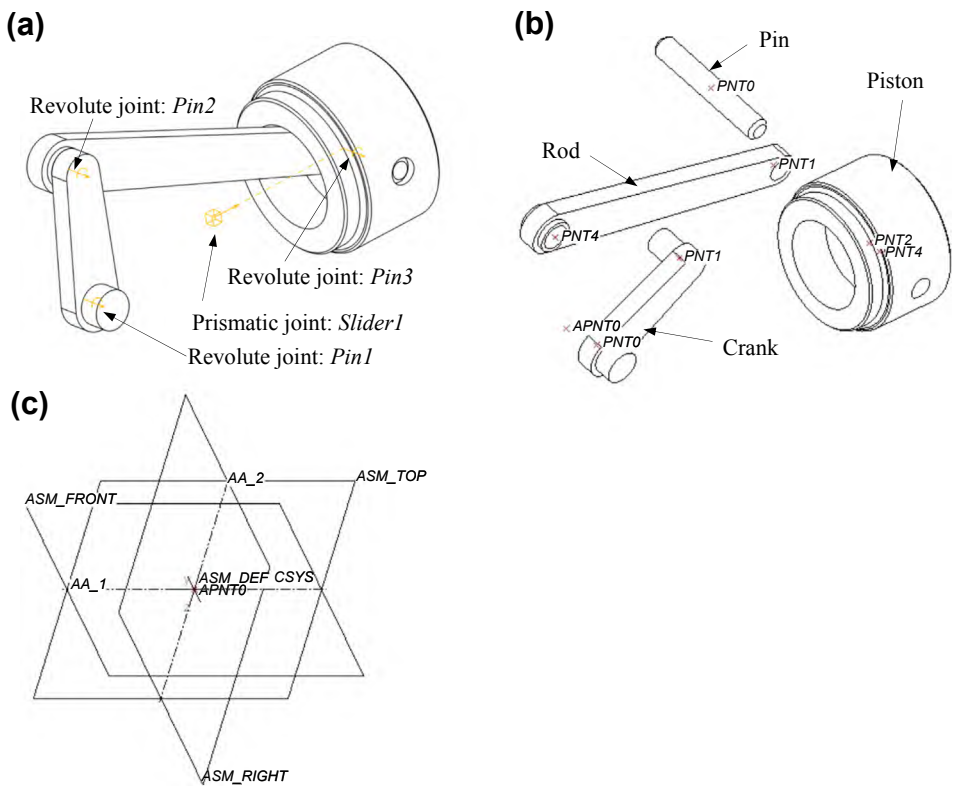


FIGURE 4.11

The slider-crank mechanism. (a) Assembled kinematic model, (b) exploded view with datum points for defining joint locations, and (c) assembly datum features serving as the ground part.

datum axes (such as *AA_1* of assembly shown in Figure 4.11(c)) created in parts and assembly will be used to define joints between parts.

The assembly datum features shown in Figure 4.11(c) include datum planes, datum axes, and datum points. Note that the datum axes *AA_1* and *AA_2* and datum point *APNT0* will be used for creating joints—specifically, the pin joint between the ground and the crank, as well as the slider joint between the ground and the piston.

We define a pin joint (*Pin1*) that allows one rotational motion between the crank and the ground. The second pin joint (*Pin2*) is created to allow rotation motion between the crank and the rod. After assembling the crank and the rod, the system should have two DOFs, allowing the crank and rod to rotate along their respective pin joints independently.

Next, the pin is assembled to the rod rigidly using placement constraints, still maintaining two DOFs. Then, the piston is assembled to the pin by defining a third pin joint. Therefore, the piston will be free to rotate along the common axes *A_1* (pin) and *A_5* (piston). The total number of DOFs now increases to three.

Finally, the piston is assembled to the ground by defining a prismatic joint. The prismatic joint is created by aligning two parallel axes (*A_6* in piston and *AA_1* in the assembly) and two datum planes (*DTM3* in piston and *ASM_TOP* in the assembly). The prismatic joint allows only one translational movement between piston and ground—that is, along the common axes without rotation. The slider-crank mechanism is now restricted to planar motion, with three rotations (*Pin1, Pin2, and Pin3*) and one translation (*Slider1*) motion. However, all three rotations and the translational motion are coupled to form a closed-loop mechanism, leaving only one free DOF, which can be any one of the three rotations or the translational motion. Note the joint symbols of Pro/ENGINEER shown in Figure 4.11(a).

The total number of DOFs of the slider-crank mechanism can also be calculated as follows by using Gruebler's count:

3 (Moving bodies) $\times$ 6 (DOFs/body) $-$ 3 (revolute joints) $\times$ 5 (DOFs/revolute) $-$ 1 (prismatic joint) $\times$ 5 (DOFs/prismatic) $= 18 - 20 = -2$.

We know that for this slider-crank mechanism there is only one DOF. However, the count yields -2. This is because there are three redundant DOFs created in the system. This is fine because the CAD system, such as Pro/ENGINEER, filters out the redundant DOFs for kinematic analysis. Joints defined in this simulation model are summarized in Table 4.6. The pairs of datum points and datum

Table 4.6 Joints Defined in the Simulation Model

	Ground Body	Crank	Rod/Pin	Piston
Crank	*Pin1 A_1* (crank)/ *AA_2* and *PNT0*/ *APNT0*		*Pin2 A_2* (crank)/ *A_1* (rod) and *PNT1* (crank)/*PNT4* (rod)	
Rod/pin		*Pin2 A_2* (crank)/ *A_1* (rod) and *PNT1* (crank)/*PNT4* (rod)		*Pin3 A_5* (piston)/ *A_1* (pin) and *PNT2* (piston)/*PNT0* (pin)
Piston	*Slider1 A_6* (piston)/*AA_1* and *DTM3* (piston)/ *ASM_TOP*		*Pin3 A_5* (piston)/ *A_1* (pin) and *PNT2* (piston)/*PNT0* (pin)	

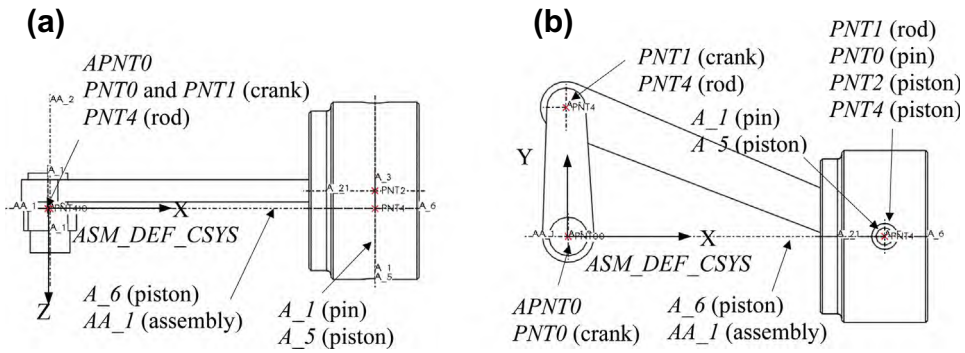


FIGURE 4.12

Locations of datum points and datum axes. (a) Top view, and (b) front view.

axes created in the parts and the assembly for defining these four joints can be seen in the top and front views of the mechanism, as shown in Figure 4.12.

Note that the way the joints are defined is not unique. One of the pin joints may be replaced with a bearing joint, which describes an identical slider-crank mechanism kinematically, in which the total DOF becomes 1.

After completing the assembly using kinematic joints, you may click the Drag Components button at the top of the graphics window in Pro/ENGINEER, and click and drag a component to see how parts move. You may also bring the assembly into Mechanism Design by choosing from the pull-down menu: Applications > Mechanism, in which you may create a driver (e.g., a rotary motor) to drive the mechanism or define a force that pushes the piston to conduct a dynamic simulation.

4.3 ASSEMBLY MODELING TECHNIQUE

An assembly model in CAD can be created by specifying assembly constraints between parts. As discussed in Section 4.2, there are mating constraints and joint constraints. In this section, we discuss the technique that determines the location and orientation of a mating part in an assembly with respect to the base part by defining mating constraints. Joint constraints will be discussed in Section 4.4.

In most mechanical assemblies, part positioning is carried out sequentially, with only two parts (or subassemblies) involved at a time. Using this strategy, a smaller number of relations, and hence constraints, must be satisfied at each stage, even for a large assembly. This can offer significant computational advantages in comparison with a simultaneous strategy.

Part positioning in assembly involves specifying part location and orientation. It can be expressed relative to some global reference or with respect to other parts. In either case, part location and orientation are specified by a 4×4 homogeneous transformation matrix. In this section, we first discuss the transformation method and solution scheme proposed by Kim et al. (2000) in Section 4.3.1. Then, we introduce a technique for degree of freedom analysis based on the mating constraints in Section 4.3.2. Again, we adopt the terminologies of mating constraints defined in SolidWorks.

4.3.1 TRANSFORMATION MATRIX

The method we discuss in this subsection takes well-constrained mating conditions between a base and a mating part and directly transforms them into a 4×4 matrix that determines the relative location and orientation of the mating part with respect to the base part. Well-constrained mating conditions imply that mating constraints are not in conflict in positioning the mating part to the base part.

In the example shown in Figure 4.13, a mating part (Figure 4.13(b)) is assembled first to the base part (Figure 4.13(a)), with a concentric mate applied to the inner surface of the hole in the base part and the cylindrical surface of the mating part (Figure 4.13(c)), in which axes of the hole and cylinder align and the mating part is free to rotate and translate along the common axis. Then, a coincident-aligned is applied to the top face of the base part and bottom face of the mating part, resulting in a fully constrained assembly (Figure 4.13(d)). To assemble the mating part to the base part, a 4×4 matrix (similar to that in Chapter 2), which is determined by directly computing a rotation matrix $\mathbf{T}_R$ and a translation matrix $\mathbf{T}_L$ that define the relative orientation and location of the mating part, respectively, must be calculated.

In determining the transformation matrix, we compute the rotation matrix $\mathbf{T}_R$ first by solving a set of linear constraint equations associated with the orientation of two mating parts. After orienting the mating part by applying the rotation matrix $\mathbf{T}_R$, the translation matrix $\mathbf{T}_L$ is calculated by solving a set of linear constraint equations associated with location. This method is computationally very effective because the transformation matrix for relative location and orientation of the mating part is algebraically derived directly from the linear equations associated with the mating conditions. We assume that the mating part is fully constrained.

The mating conditions considered in this subsection are concentric and coincident. We adopt the conventions in Kim et al. (2005), in which the superscripts *b*, *m*, *mr*, and *ma* in the following equations indicate the base part, the mating part, the mating part after rotation, and the mating part after assembly, respectively.

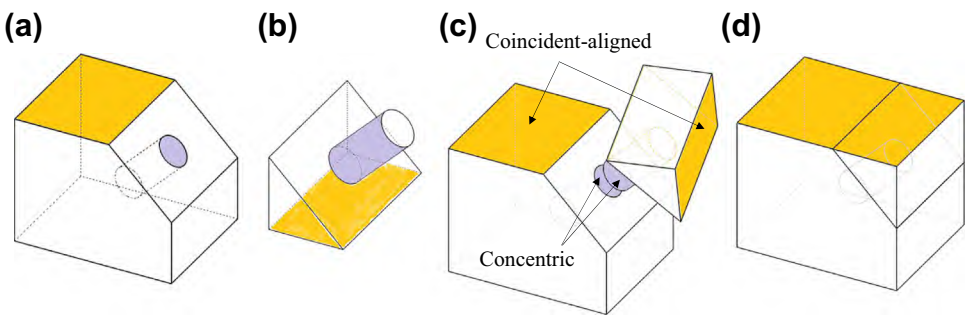


FIGURE 4.13

An example of a two-part assembly. (a) Base part, (b) mating part, (c) concentric mating constraint applied to the hole in the base part and the cylindrical surface in the mating part, and (d) coincident-aligned applied to the top face of the base part and bottom face of the mating part, resulting in a fully constrained assembly.

4.3.1.1 Coincident

The coincident-mate holds between two planar faces and requires the two faces to touch each other (Figure 4.14(a)). The designated faces, shaded in Figure 4.14(a), are the faces to be mated. Each face is specified by its unit normal vector $\mathbf{n}$ and one point $\mathbf{P}$ on the face in terms of its local coordinate system. This condition is accomplished by constraining the two normal vectors to be opposite to each other, and the two points that are noncoincident to lie on the same plane at which the two faces mate. Thus, equations of the coincident-mate constraint can be expressed as follows:

$$\mathbf{n}^b = -\mathbf{n}^{ma} \tag{4.1a}$$

where $\mathbf{n}^b = [\mathrm{n}_x^b, \mathrm{n}_y^b, \mathrm{n}_z^b]^{\mathrm{T}}$ and $\mathbf{n}^{ma} = [\mathrm{n}_x^{ma}, \mathrm{n}_y^{ma}, \mathrm{n}_z^{ma}]^{\mathrm{T}}$ are also called direction vectors; and

$$\mathbf{n}^{b^{\mathrm{T}}} \cdot (\mathbf{P}^b - \mathbf{P}^{ma}) = 0 \tag{4.1b}$$

where $\mathbf{P}^b = [\mathrm{P}_x^b, \mathrm{P}_y^b, \mathrm{P}_z^b]^{\mathrm{T}}$, and $\mathbf{P}^{ma} = [\mathrm{P}_x^{ma}, \mathrm{P}_y^{ma}, \mathrm{P}_z^{ma}]^{\mathrm{T}}$. Note that $\mathbf{P}^b$ and $\mathbf{P}^{ma}$ must not coincide.

The coincident-aligned condition is assigned between two planar faces when they lie in the same plane, as shown in Figure 4.14(b). Equations of the coincident-aligned constraint are similar to those of the coincident-mate constraint, except that the two normal vectors $\mathbf{n}^b$ and $\mathbf{n}^m$ are required to be in the same direction. Thus, a coincident-aligned constraint can be expressed mathematically by

$$\mathbf{n}^b = \mathbf{n}^{ma} \tag{4.2a}$$

and

$$\mathbf{n}^{b^{\mathrm{T}}} \cdot (\mathbf{P}^b - \mathbf{P}^{ma}) = 0. \tag{4.2b}$$

4.3.1.2 Concentric

The concentric condition holds between two cylindrical faces: a shaft face, and a hole face, as shown in Figures 4.14(c) and (d). The concentric condition is accomplished by requiring the center axes of shaft and hole components to be parallel and a point $\mathbf{P}^m$ on the axis of the mating part lies on the axis of the base part. An axis is defined by a unit direction vector and a point on it. The hole axis is specified by a point $\mathbf{P}^b$ and a unit direction vector $\mathbf{n}^b$ defined in terms of its local coordinate system. Similarly, the shaft axis is specified by a point $\mathbf{P}^m$ and a unit direction vector $\mathbf{n}^m$ in terms of its local coordinate system. Thus, the constraint equations for concentric conditions can be written as

$$\mathbf{n}^b = \mathbf{n}^{ma} \tag{4.3a}$$

for aligned (see Figure 4.14(c)), $\mathbf{n}^b = -\mathbf{n}^{ma}$ for antialigned (see Figure 4.14(d)), and

$$\frac{\mathrm{P}_x^{ma} - \mathrm{P}_x^b}{\mathrm{n}_x^b} = \frac{\mathrm{P}_y^{ma} - \mathrm{P}_y^b}{\mathrm{n}_y^b} = \frac{\mathrm{P}_z^{ma} - \mathrm{P}_z^b}{\mathrm{n}_z^b} = \mathrm{C} \neq 0 \tag{4.3b}$$

because vectors $\mathbf{n}^b$ and $\mathbf{P}^{ma} - \mathbf{P}^b$ are collinear.

Equations 4.1–4.3 specify partially the relative rotation and translation of the mating part with respect to the base part, associated with the respective mating constraints. We assume that the origins of the coordinate systems of the mating and base parts coincide (not necessarily aligned) before applying the mating constraints, although they are sketched separately in Figure 4.14 for clarity. This point is illustrated in Example 4.1.

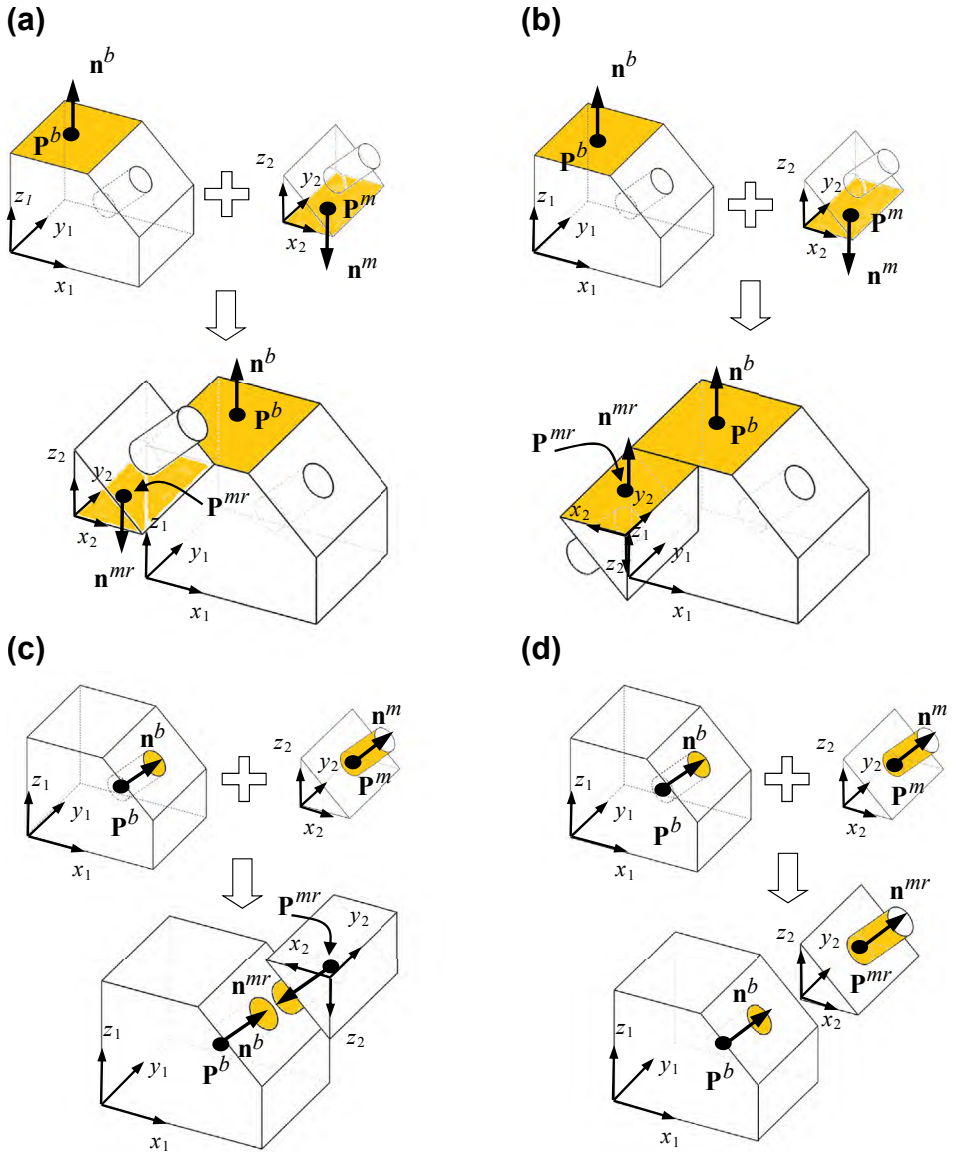


FIGURE 4.14

Assembly mates. (a) Coincident-mate, (b) coincident-aligned, (c) concentric-mate, and (d) concentric-aligned.

Although we only present equations for concentric and coincident mating constraints, equations of remaining mating constraints can be derived following the same ideas presented. For example, Eq. 4.1a or Eq. 4.2a (for mate or align, respectively) is sufficient to support a parallel mating constraint. In addition, a coincident offset constraint can be represented by using the same equation as either Eq. 4.1a or Eq. 4.2a (for mate or align, respectively) for part orientation, and the following equation for location:

$$\mathbf{n}^{b^{\mathrm{T}}} \cdot \left[\mathbf{P}^{b} - (\mathbf{P}^{ma} - \mathbf{O})\right] = 0 \tag{4.3c}$$

where $\mathbf{O}$ is the vector of the offset specified by the designer.

4.3.1.3 Computation of the Transformation Matrix

The relative orientation and location of the mating part with respect to the base part is represented by a 4×4 transformation matrix. The transformation matrix can be written in homogeneous coordinates, defined as

$$\mathbf{T} = \begin{bmatrix} R_{1x} & R_{1y} & R_{1z} & L_x \\ R_{2x} & R_{2y} & R_{2z} & L_y \\ R_{3x} & R_{3y} & R_{3z} & L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{L} \\ 0 & 1 \end{bmatrix} \tag{4.4a}$$

in which the 3×3 matrix $\mathbf{R}$ defines the rotation transformation, and the 3×1 column vector $\mathbf{L} = [L_x\, L_y\, L_z]^{\mathrm{T}}$ determines the translation. Physically, this transformation can be viewed as a representation of a coordinate system in a fixed reference coordinate system. Each unit vector of the coordinate frame $\mathbf{n}_1$, $\mathbf{n}_2$, and $\mathbf{n}_3$ is mutually perpendicular, as illustrated in Figure 4.15. With this, the transformation matrix can be rewritten as

$$\mathbf{T} = \begin{bmatrix} \mathbf{n}_{1x} & \mathbf{n}_{2x} & \mathbf{n}_{3x} & \mathrm{L}_x \\ \mathbf{n}_{1y} & \mathbf{n}_{2y} & \mathbf{n}_{3y} & \mathrm{L}_y \\ \mathbf{n}_{1z} & \mathbf{n}_{2z} & \mathbf{n}_{3z} & \mathrm{L}_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{L} \\ 0 & 1 \end{bmatrix}. \tag{4.4b}$$

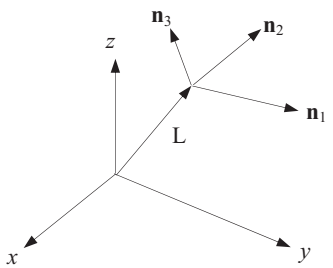


FIGURE 4.15

Representation of a frame in a frame.

With such a transformation matrix, any given vector $\mathbf{v} = [v_x\ v_y\ v_z]^{\mathrm{T}}$ defined in the mating part with respect to its local coordinate system can be transformed to $\mathbf{V} = [V_x\ V_y\ V_z]^{\mathrm{T}}$ with respect to the coordinate system of the base part by the following,

$$\begin{bmatrix} \mathbf{V} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{L} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{Rv} + \mathbf{L} \\ 1 \end{bmatrix} \tag{4.5}$$

which clearly shows the rotational and translational portions of the transformation. Therefore, the transformation matrix $\mathbf{T}$ in Eq. 4.4 can be represented by the product of a translation matrix $\mathbf{T}_{\mathrm{L}}$ and a rotation matrix $\mathbf{T}_{\mathrm{R}}$ as

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & \mathbf{L} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{T}_{\mathrm{L}}\mathbf{T}_{\mathrm{R}} \tag{4.6a}$$

where

$$\mathbf{T}_{\mathrm{R}} = \begin{bmatrix} \mathbf{R} & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{1x} & R_{1y} & R_{1z} & 0 \\ R_{2x} & R_{2y} & R_{2z} & 0 \\ R_{3x} & R_{3y} & R_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6b}$$

and

$$\mathbf{T}_{\mathrm{L}} = \begin{bmatrix} \mathbf{I} & \mathbf{L} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & L_x \\ 0 & 1 & 0 & L_y \\ 0 & 0 & 1 & L_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.6c}$$

These matrices are determined sequentially. We first derive $\mathbf{T}_{\mathrm{R}}$ from the rotational relationships between the mating parts, and then derive $\mathbf{T}_{\mathrm{L}}$ from the translational relationships between the base part and the mating part after it is reoriented by applying the rotation matrix $\mathbf{T}_{\mathrm{R}}$.

When we are given two independent pairs of direction vectors, $(\mathbf{n}_i^b, \mathbf{n}_i^m)$, $i = 1, 2$, from the well-constrained mating conditions between a base part and a mating part, as shown in Figure 4.16, the equation associated with rotation of components is expressed as

$$\mathbf{n}_i^{mr} = \mathbf{R}\,\mathbf{n}_i^m, \quad i = 1, 2 \tag{4.7}$$

where $\mathbf{n}_i^{mr} = \mathbf{n}_i^b$, $i = 1, 2$ is for aligned and $\mathbf{n}_i^{mr} = -\mathbf{n}_i^b$, $i = 1, 2$ is for mate.

Here, $\mathbf{n}_i^{mr}$ is a mating direction vector after the mating part is reoriented by applying the rotation matrix $\mathbf{T}_{\mathrm{R}}$. If $\mathbf{n}_3^m$ and $\mathbf{n}_3^{mr}$ are defined as $\mathbf{n}_3^m = \mathbf{n}_1^m \times \mathbf{n}_2^m$ and $\mathbf{n}_3^{mr} = \mathbf{n}_1^{mr} \times \mathbf{n}_2^{mr}$, respectively, then, the relation between $\mathbf{n}_3^m$ and $\mathbf{n}_3^{mr}$ is derived as $\mathbf{n}_3^{\mathrm{mr}} = \mathbf{R}\,\mathbf{n}_3^{\mathrm{m}}$. These equations are rewritten as the matrix product

$$[\,\mathbf{n}_1^{mr} \quad \mathbf{n}_2^{mr} \quad \mathbf{n}_3^{mr}\,] = \mathbf{R}[\,\mathbf{n}_1^m \quad \mathbf{n}_2^m \quad \mathbf{n}_3^m\,]. \tag{4.8a}$$

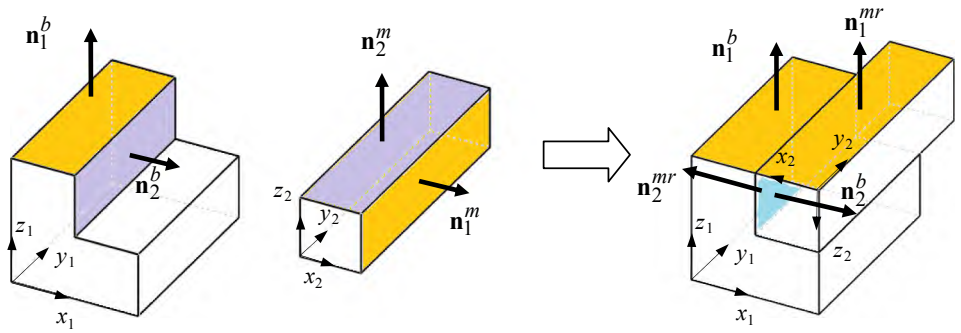


FIGURE 4.16

Mating conditions for assembly modeling.

Hence, the rotational submatrix $\mathbf{R}$ is obtained by

$$\mathbf{R} = [\,\mathbf{n}_1^{mr} \quad \mathbf{n}_2^{mr} \quad \mathbf{n}_3^{mr}\,][\,\mathbf{n}_1^{m} \quad \mathbf{n}_2^{m} \quad \mathbf{n}_3^{m}\,]^{-1}. \tag{4.8b}$$

The translation submatrix $\mathbf{L}$ is computed algebraically by solving the equations associated with translation after reorienting the mating part by applying the rotation matrix $\mathbf{T}_{\mathrm{R}}$. After reorienting, the mating direction vectors are parallel and a point on the mating part after assembly $\mathbf{P}^{ma}$ is expressed as

$$\mathbf{P}^{ma} = \mathbf{P}^{mr} + \mathbf{L} \tag{4.9}$$

where $\mathbf{P}^{mr}$ is a point on the mating part after reorienting and is obtained by $\mathbf{P}^{mr} = \mathbf{R}\cdot\mathbf{P}^{m}$. Thus, the constraint equations associated with the translation of mating parts are expressed next.

First, we consider coincident-mate, in which the direction vectors of the mating parts are parallel after reorientation. The condition requires that one point on the mating face of the mating part lies on the mating face of the base part and they are not coincident. Thus, the translational constraint equation for the coincident-mate is expressed, following Eq. 4.1b, as:

$$\mathbf{n}^{b^{\mathrm{T}}}\cdot(\mathbf{P}^{b} - \mathbf{P}^{ma}) = \mathbf{n}^{b^{\mathrm{T}}}\cdot\left[\mathbf{P}^{b} - (\mathbf{P}^{mr} + \mathbf{L})\right] = 0. \tag{4.10}$$

For concentric mate, we know that the center axes of the mating parts are parallel after repositioning; the constraint requires that one point $\mathbf{P}^{m}$ on the axis of the mating part lies on the axis of the base part. Thus, the translational constraint equations for the concentrate mate are expressed, following Eq. 4.3b, as:

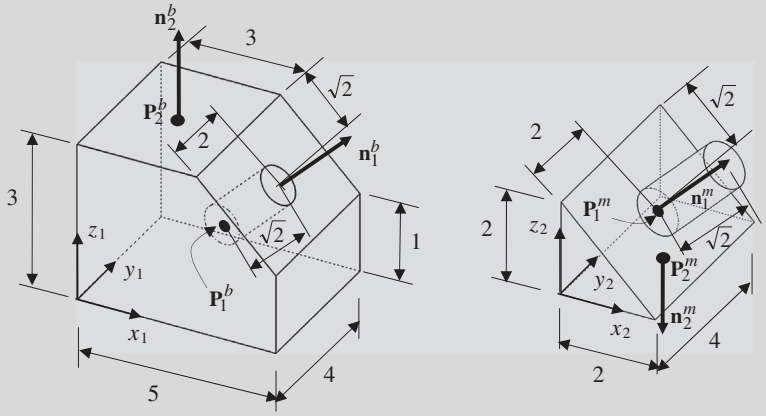
$$\frac{(\mathrm{P}_{1x}^{mr} + \mathrm{L}_x) - \mathrm{P}_{1x}^{b}}{\mathbf{n}_{1x}^{b}} = \frac{(\mathrm{P}_{1y}^{mr} + \mathrm{L}_y) - \mathrm{P}_{1y}^{b}}{\mathbf{n}_{1y}^{b}} = \frac{(\mathrm{P}_{1z}^{mr} + \mathrm{L}_z) - \mathrm{P}_{1z}^{b}}{\mathbf{n}_{1z}^{b}} = \mathrm{C} \neq 0 \tag{4.11a}$$

or

$$\begin{aligned}
\left(\mathrm{P}_{1x}^{mr} + \mathrm{L}_x\right) - \mathrm{P}_{1x}^{b} &= \mathrm{Cn}_{1x}^{b} \\
\left(\mathrm{P}_{1y}^{mr} + \mathrm{L}_y\right) - \mathrm{P}_{1y}^{b} &= \mathrm{Cn}_{1y}^{b} \\
\left(\mathrm{P}_{1z}^{mr} + \mathrm{L}_z\right) - \mathrm{P}_{1z}^{b} &= \mathrm{Cn}_{1z}^{b}.
\end{aligned} \tag{4.11b}$$

EXAMPLE 4.1

Consider the two assembly components shown in Figure 4.13. Each component is defined in its local coordinate system, with dimensions shown in the figures below. The two components are assembled by applying a concentric mate to the inner surface of the hole in the base part and the outer surface of the cylinder in the mating part, and imposing a coincident-aligned on the top face of the base part and the bottom face of the mating part, as discussed before. Therefore, the vectors $\mathbf{n}_1^b$ and $\mathbf{n}_1^m$ align with the axes of the hole and cylinder, respectively, and the vectors $\mathbf{n}_2^b$ and $\mathbf{n}_2^m$ are normal to the faces on the respective components, as shown below. The reference point $\mathbf{P}_1^b$ is at the center of the bottom face of the hole, and point $\mathbf{P}_1^m$ is located at the center of the bottom face of the cylinder on the mating part. Also, reference points $\mathbf{P}_2^b$ and $\mathbf{P}_2^m$ are located at the center of their respective mate planes, as shown in the figures below. These points must be chosen such that they do not coincide after assembly. The table below lists the geometric data of the vectors and reference points.



	Base Part	Mating Part	Mating Part After Assembly
Concentric	$\mathbf{n}_1^b = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \mathbf{P}_1^b = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$	$\mathbf{n}_1^m = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \mathbf{P}_1^m = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$	$\mathbf{n}_1^{mr} = -\mathbf{n}_1^b = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ 0 \\ -\frac{1}{\sqrt{2}} \end{bmatrix}, \mathbf{P}_1^{ma} = \begin{bmatrix} 4 \\ 2 \\ 2 \end{bmatrix}$
Coincident	$\mathbf{n}_2^b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \mathbf{P}_2^b = \begin{bmatrix} 1.5 \\ 2 \\ 3 \end{bmatrix}$	$\mathbf{n}_2^m = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \mathbf{P}_2^m = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$	$\mathbf{n}_2^{mr} = \mathbf{n}_2^b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \mathbf{P}_2^{ma} = \begin{bmatrix} 4 \\ 2 \\ 3 \end{bmatrix}$

When the two direction vectors $\mathbf{n}_1^m$ and $\mathbf{n}_2^m$ are given, the third direction vector can be computed as

$$\mathbf{n}_3^m = \mathbf{n}_1^m \times \mathbf{n}_2^m = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}.$$

Continued

EXAMPLE 4.1—cont'd

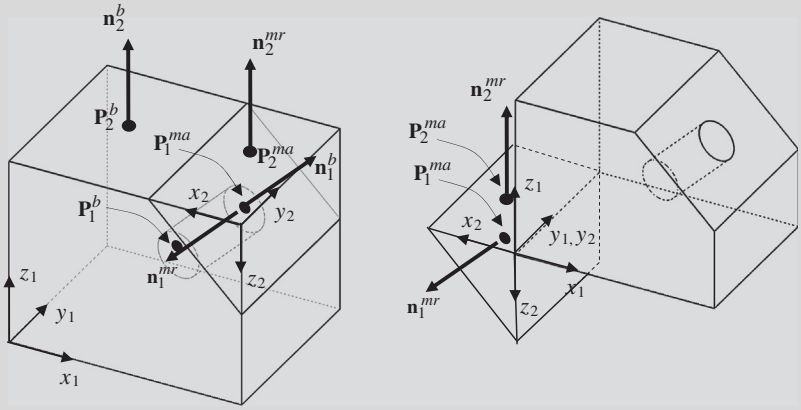
After normalizing it, we have $\mathbf{n}_3^m = [0 \quad 1 \quad 0]^{\mathrm{T}}$. After applying these two mating constraints, the mating part is assembled to the base part shown in the figure below (left). The vectors and reference points after assembly are listed in the table above.

Therefore, we have

$$\mathbf{n}_3^{mr} = \mathbf{n}_1^{mr} \times \mathbf{n}_2^{mr} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ 0 \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}.$$

After normalizing it, we have $\mathbf{n}_3^m = [0 \quad 1 \quad 0]^{\mathrm{T}}$. Then the rotation matrix $\mathbf{R}$ can be calculated using Eq. 4.8b as

$$\mathbf{R} = \begin{bmatrix} \mathbf{n}_1^{mr} & \mathbf{n}_2^{mr} & \mathbf{n}_3^{mr} \end{bmatrix} \begin{bmatrix} \mathbf{n}_1^{m} & \mathbf{n}_2^{m} & \mathbf{n}_3^{m} \end{bmatrix}^{-1} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{\sqrt{2}} & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 \\ \frac{1}{\sqrt{2}} & -1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$



Note that the rotation matrix $\mathbf{R}$ rotates the mating part to an orientation shown in the figure above (right). The columns of the submatrix $\mathbf{R}$ represent respectively the three coordinate axes of the coordinate system in the mating part with respect to the coordinate system of the base part. For example, the first column of the submatrix $\mathbf{R}$ shows that the axis x_2 is now aligned with x_1 but in the opposite direction.

Now we find the translation matrix $\mathbf{L}$. Using the rotation submatrix $\mathbf{R}$ obtained in the previous step, two points $\mathbf{P}_1^{mr}$ and $\mathbf{P}_2^{mr}$ on the reoriented mating part are computed first by

$$\mathbf{P}_1^{mr} = \mathbf{R}\mathbf{P}_1^m = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$$

EXAMPLE 4.1—cont'd

and

$$\mathbf{P}_2^{mr} = \mathbf{R}\mathbf{P}_2^{m} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix}.$$

Then, the translational constraint equations are derived from the mating conditions as follows. From the coincident condition in Eq. 4.10, we have

$$\mathbf{n}_2^{b^{\mathrm{T}}} \cdot \left[\mathbf{P}_2^{b} - \left(\mathbf{P}_2^{mr} + \mathbf{L} \right) \right] = 0, \quad \text{or} \quad [0 \quad 0 \quad 1] \left\{ \begin{bmatrix} 1.5 \\ 2 \\ 3 \end{bmatrix} - \left(\begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix} \right) + \begin{bmatrix} L_x \\ L_y \\ L_z \end{bmatrix} \right\} = 0$$

which gives $L_z = 3$. From the concentric condition in Eq. 4.11a, we have

$$\frac{\left(\mathrm{P}_{1x}^{mr} + L_x \right) - \mathrm{P}_{1x}^{b}}{\mathrm{n}_{1x}^{b}} = \frac{\left(\mathrm{P}_{1y}^{mr} + \mathrm{L}_y \right) - \mathrm{P}_{1y}^{b}}{\mathrm{n}_{1y}^{b}} = \frac{\left(\mathrm{P}_{1z}^{mr} + \mathrm{L}_z \right) - \mathrm{P}_{1z}^{b}}{\mathrm{n}_{1z}^{b}} = \mathrm{C} \neq 0$$

or

$$\frac{(-1 + L_x) - 3}{\frac{1}{\sqrt{2}}} = \frac{(2 + L_y) - 2}{0} = \frac{(-1 + L_z) - 1}{\frac{1}{\sqrt{2}}} = \frac{(-1 + 3) - 1}{\frac{1}{\sqrt{2}}} = \frac{1}{\frac{1}{\sqrt{2}}} = \mathrm{C}.$$

Hence, $L_x = 5$, and $L_y = 0$. Therefore, the translational submatrix is $\mathbf{L} = [5, 0, 3]^{\mathrm{T}}$.

Thus, the transformation matrix $\mathbf{T}$ for relative orientation and location of the mating part with respect to the base part is obtained as

$$\mathbf{T} = \begin{bmatrix} R_{1x} & R_{1y} & R_{1z} & L_x \\ R_{2x} & R_{2y} & R_{2z} & L_y \\ R_{3x} & R_{3y} & R_{3z} & L_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

One of the potential pitfalls in using the method discussed above for representing an assembly is that the rotation matrix derived from Eq. 4.8b may not be orthogonal. To carry out a valid rotation, the rotation matrix must satisfy the two basic properties of orthogonality: $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$ and $|\mathbf{R}| = 1$.

We use the following example to illustrate the pitfall.

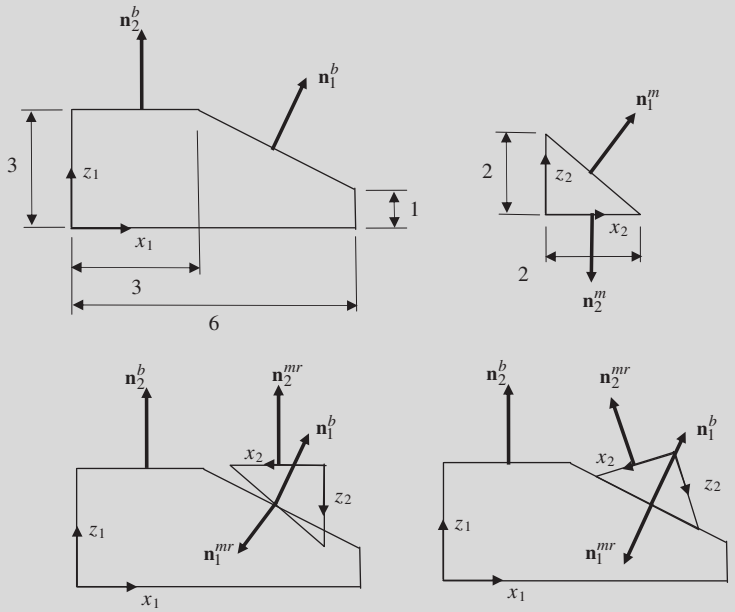
The orthogonality properties of a rotation matrix can be easily verified by computer. When the properties are not satisfied, the mating constraints are in conflict, and CAD prompts an error message.

EXAMPLE 4.2

Consider the same assembly as in Example 4.1 except that the length of the base part is increased from 5 to 6, as shown in the figure below (top left). We only show side views with direction vectors to get to the points of this example.

It is apparent that the two mating constraints are in conflict in terms of determining an orientation of the mating part that satisfies both constraints. You may either impose a coincident-aligned on the top face of the base part and the bottom face of the mating part (shown in the lower left figure below), in which the vectors $\mathbf{n}_2^b$ and $\mathbf{n}_2^m$ align, or apply a concentric mate to the inner surface of the hole in the base part and the outer surface of the cylinder in the mating part (shown in the lower right figure below), in which the vectors $\mathbf{n}_1^b$ and $\mathbf{n}_1^m$ align. It is impossible to orient the mating part so that both sets of the normal vectors align simultaneously.

With this understanding, we will proceed with computing the rotation matrix $\mathbf{R}$ following the steps discussed and then point out the pitfall.



We first assume that both constraints are satisfied; hence, $\mathbf{n}_1^{mr} = -\mathbf{n}_1^b$ and $\mathbf{n}_2^{mr} = \mathbf{n}_2^b$. The table below lists the geometric data of the vectors and reference points.

	Base Part	**Mating Part**	**Mating Part After Assembly**
Concentric	$\mathbf{n}_1^b = \begin{bmatrix} \frac{2}{\sqrt{13}} \\ 0 \\ \frac{3}{\sqrt{13}} \end{bmatrix}$	$\mathbf{n}_1^m = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}$	$\mathbf{n}_1^{mr} = -\mathbf{n}_1^b = \begin{bmatrix} -\frac{2}{\sqrt{13}} \\ 0 \\ -\frac{3}{\sqrt{13}} \end{bmatrix}$
Coincident	$\mathbf{n}_2^b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\mathbf{n}_2^m = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$	$\mathbf{n}_2^{mr} = \mathbf{n}_2^b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$

EXAMPLE 4.2—cont'd

With the two direction vectors $\mathbf{n}_1^{\mathrm{m}}$ and $\mathbf{n}_2^{\mathrm{m}}$ given, the third direction vector can be computed as

$$\mathbf{n}_3^m = \mathbf{n}_1^m \times \mathbf{n}_2^m = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix}.$$

After normalizing it, we have $\mathbf{n}_3^m = [0 \quad 1 \quad 0]^{\mathrm{T}}$.

Similarly, the third vector after assembly is

$$\mathbf{n}_3^{mr} = \mathbf{n}_1^{mr} \times \mathbf{n}_2^{mr} = \begin{bmatrix} -\frac{2}{\sqrt{13}} \\ 0 \\ -\frac{3}{\sqrt{13}} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{2}{\sqrt{13}} \\ 0 \end{bmatrix}.$$

After normalizing it, we have $\mathbf{n}_3^{mr} = [0 \quad 1 \quad 0]^{\mathrm{T}}$.

Then the rotation matrix $\mathbf{R}$ can be calculated using Eq. 4.8 as

$$\mathbf{R} = \begin{bmatrix} \mathbf{n}_1^{mr} & \mathbf{n}_2^{mr} & \mathbf{n}_3^{mr} \end{bmatrix} \begin{bmatrix} \mathbf{n}_1^{m} & \mathbf{n}_2^{m} & \mathbf{n}_3^{m} \end{bmatrix}^{-1} = \begin{bmatrix} -\frac{2}{\sqrt{13}} & 0 & 0 \\ 0 & 0 & 1 \\ -\frac{3}{\sqrt{13}} & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 1 \\ \frac{1}{\sqrt{2}} & -1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} -\frac{2\sqrt{2}}{\sqrt{13}} & 0 & 0 \\ 0 & 1 & 0 \\ 1-\frac{3\sqrt{2}}{\sqrt{13}} & 0 & -1 \end{bmatrix}.$$

If we rotate the two direction vectors of the mating part before assembly, we have

$$\mathbf{n}_1^{mr} = \mathbf{R}\mathbf{n}_1^m = \begin{bmatrix} -\frac{2\sqrt{2}}{\sqrt{13}} & 0 & 0 \\ 0 & 1 & 0 \\ 1-\frac{3\sqrt{2}}{\sqrt{13}} & 0 & -1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} -\frac{2}{\sqrt{13}} \\ 0 \\ -\frac{3}{\sqrt{13}} \end{bmatrix}$$

which is $-\mathbf{n}_1^b$, as it should be, and

$$\mathbf{n}_2^{mr} = \mathbf{R}\mathbf{n}_2^m = \begin{bmatrix} -\frac{2\sqrt{2}}{\sqrt{13}} & 0 & 0 \\ 0 & 1 & 0 \\ 1-\frac{3\sqrt{2}}{\sqrt{13}} & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

which is $\mathbf{n}_2^b$. Everything seems to be working fine mathematically. However, we know at the beginning that such a transformation is impossible physically. What is the problem? Let us take a look at the transformation matrix $\mathbf{R}$. First,

$$\mathbf{R}\mathbf{R}^{\mathrm{T}} = \begin{bmatrix} -\frac{2\sqrt{2}}{\sqrt{13}} & 0 & 0 \\ 0 & 1 & 0 \\ 1-\frac{3\sqrt{2}}{\sqrt{13}} & 0 & -1 \end{bmatrix} \begin{bmatrix} -\frac{2\sqrt{2}}{\sqrt{13}} & 0 & 1-\frac{3\sqrt{2}}{\sqrt{13}} \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} = \begin{bmatrix} \frac{8}{13} & 0 & -\frac{2\sqrt{2}}{\sqrt{13}}+\frac{12}{13} \\ 0 & 1 & 0 \\ -\frac{2\sqrt{2}}{\sqrt{13}}+\frac{12}{13} & 0 & -\frac{6\sqrt{2}}{\sqrt{13}}+\frac{44}{13} \end{bmatrix} \neq \mathbf{I}$$

Continued

EXAMPLE 4.2—cont'd

and

$$|\mathbf{R}| = \begin{bmatrix} -\frac{2\sqrt{2}}{\sqrt{13}} & 0 & 0 \\ 0 & 1 & 0 \\ 1-\frac{3\sqrt{2}}{\sqrt{13}} & 0 & -1 \end{bmatrix} = 0.7845 \neq 1.$$

Therefore, the rotation submatrix $\mathbf{R}$ is not orthogonal.

The problem with a nonorthogonal rotation matrix is that it does not perform the rotation correctly. For example, if we rotate the vectors that represent respectively axes x_2 and z_2 using the submatrix $\mathbf{R}$, we have

$$\mathbf{n}_{x_2}^{mr} = \mathbf{R}\mathbf{n}_{x_2}^{m} = \begin{bmatrix} -\frac{2\sqrt{2}}{\sqrt{13}} & 0 & 0 \\ 0 & 1 & 0 \\ 1-\frac{3\sqrt{2}}{\sqrt{13}} & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -\frac{2\sqrt{2}}{\sqrt{13}} \\ 0 \\ 1-\frac{3\sqrt{2}}{\sqrt{13}} \end{bmatrix}$$

and

$$\mathbf{n}_{z_2}^{mr} = \mathbf{R}\mathbf{n}_{z_2}^{m} = \begin{bmatrix} -\frac{2\sqrt{2}}{\sqrt{13}} & 0 & 0 \\ 0 & 1 & 0 \\ 1-\frac{3\sqrt{2}}{\sqrt{13}} & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}.$$

These two vectors are nonperpendicular:

$$\mathbf{n}_{x_2}^{mr} \cdot \mathbf{n}_{z_2}^{mr} = \begin{bmatrix} -\frac{2\sqrt{2}}{\sqrt{13}} & 0 & 1-\frac{3\sqrt{2}}{\sqrt{13}} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = -1 + \frac{3\sqrt{2}}{\sqrt{13}} \neq 0$$

which is wrong, to say the least.

In practice, a product assembly must be properly parameterized so that a desired design intent is captured without invoking any conflict between mating constraints. In this example, there are two obvious possibilities. One intent is keeping the slope of the mating surfaces as 45°, as shown in Figure 4.17(b). In this case, the width of the top edge of the base part must be related to that of the bottom edge, such as $d_{top}^{b} = d_{bottom}^{b} - 2$, where d_{top}^{b} and d_{bottom}^{b} are the widths of the top and bottom edges of the base part, respectively. The second intent, as illustrated in Figure 4.17(b), allows the slope of the mating surfaces to vary. In this case, the widths of the top and bottom edges of the base part must be related to the bottom edge of the mating part, such as $d_{bottom}^{m} = d_{bottom}^{b} - d_{top}^{b}$, where d_{bottom}^{m} is the width of the bottom edge of the mating part.

The assembly modeling approach discussed is capable of supporting both cases, as long as there is no conflict between the mating constraints. For the first intent shown in Figure 4.17(a), the transformation matrix can be calculated following the same steps shown in Example 4.1, in which point $\mathbf{P}_1^b$ is relocated according to the dimension d_{bottom}^{b}. The resulting transformation matrix is identical to that of Example 4.1, except for L_x, which is determined by the dimension d_{bottom}^{b}.

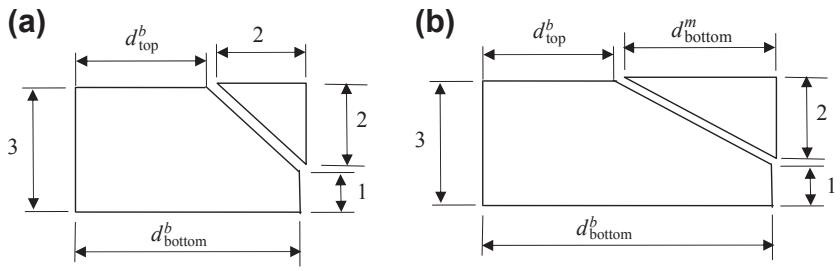


FIGURE 4.17

Illustration of design intents. (a) Slope of the mating surface kept at 45°, and (b) slope of the mating surface varying.

As for the second intent shown in Figure 4.17(b), vectors $\mathbf{n}_1^b$ and $\mathbf{n}_1^m$ as well as points $\mathbf{P}_1^b$ and $\mathbf{P}_1^m$ must be calculated, based on the change of dimension d^b_{bottom}. This is left as an exercise.

4.3.2 DEGREE OF FREEDOM ANALYSIS

In the mating constraint method, designers specify the relative positions of parts by interactively defining spatial relationships between the geometric features of mating parts. These mating constraints are applied to the same type of mating features, such as a pair of planar faces. Each of the geometry mating constraints has a pair of direction vectors, called principal vectors, which characterize the mating geometric entities. For example, the principal vectors are two outbound unit vectors that are normal to the mating planes for the coincident-mate and coincident-aligned constraints, and two unit vectors parallel to the mating axis direction for the concentric constraint. The principal vectors are antialigned for coincident-mate, whereas they are aligned for coincident-aligned.

Using these mating geometric features and principal vectors, we can determine the remaining DOFs for a pair of mating parts. For example, the two-part assembly shown in Figure 4.13 is assembled using concentric and coincident-aligned constraints as discussed. After imposing the concentric constraint between the hole in the base part and the cylindrical surface of the mating part, the two unit vectors parallel to the mating axis direction align, creating one common principal vector. At this point, the mating part is allowed to rotate along the direction of the principal vector, as shown in Figure 4.13(c). After imposing the coincident-aligned constraint between the top face of the base part and bottom face of the mating part, the two outbound unit vectors normal to the faces for the coincident-aligned constraints point in the same principal vector direction. These two principal vectors are not in parallel and are called independent. Imposing both mating constraints eliminates the rotation degree of freedom of the mating part. In general, one independent principal vector (IPV) allows one rotational DOF, and two or more IPVs eliminate all rotational DOF, as summarized in Table 4.7.

Table 4.7 Rotational DOF Analysis

Number of Independent Principal Vectors	RDOF
0	R3
1	R1
2 or more	R0

Table 4.8 Translational DOF Analysis

Intersection Mating Geometry	TDOF
Plane	T2
Line	T1
Point	T0

Note that for the bearing and crank example shown in Figure 4.3, there are two mating constraints imposed, concentric between the hole of the bearing and the cylindrical surface of the lower shaft of the crank, and a coincident-mate between the two opposite faces of the two components. These two mating constraints form two principal vectors that are parallel, counting as one IPV, therefore allowing one rotational DOF according to Table 4.7.

Using the intersection mating geometry (IMG) of mating components, we can also compute the translational DOF as shown in Table 4.8. For example, a coincident-aligned constraint yields a face IMG (Figure 4.18(a)), therefore allowing translational movement in two directions on the face. A line IMG illustrated in Figure 4.18(b) as an example allows one translational DOF along the line, which is formed by intersecting two faces normal to the two respective IPV: $\mathbf{n}_{\text{ipv1}}$ and $\mathbf{n}_{\text{ipv2}}$ in Figure 4.18(b). Figure 4.18(c) illustrates a case of point IMG, in which no translational movement is allowed. The point IMG is identified by intersecting the IPV ($\mathbf{n}_{\text{ipv1}}$ in Figure 4.18(c)) and the plane normal to the other IPV, $\mathbf{n}_{\text{ipv2}}$.

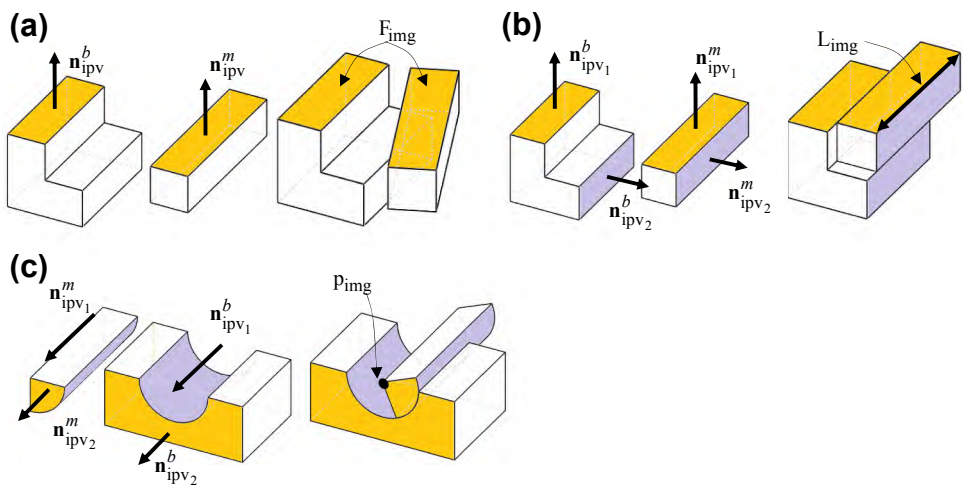


FIGURE 4.18

Examples of DOF analysis. (a) One IPV and a face IMG, (b) two IPVs and a line IMG, and (c) one IPV and a point IMG.

4.4 KINEMATIC MODELING TECHNIQUE*

The approach discussed in Section 4.3 sequentially positions the mating part with respect to the base part such that the given mating constraints are satisfied. The method is powerful and general. It is also capable of regenerating the assembly model after a design change is made as long as the mating constraints after the change are not in conflict. On top of that, the major advantage of the approach is that it solves the equations sequentially without dealing with a system of constraint equations simultaneously, as do many other methods proposed in the literature (e.g., Lee and Andrews, 1985).

However, this method requires that an assembly does not contain any closed-loop or underconstrained states, thus requiring that the mating part be positioned with respect to the base part whose position is determined. Such a limitation prevents the approach from dealing with two issues that are commonly encountered in product design involving assemblies. First, when a design change takes place, such as changing the dimensions of the crank and connecting rod for the slider-crank mechanism shown in Figure 4.19, the location and orientation of the individual parts are to be determined. Only when the location and orientation of the individual parts are determined can the method discussed in Section 4.3 be employed to calculate the transformation matrices for individual parts in the assembly. The second issue is that if the assembly is underconstrained, when a part in the assembly is moved, parts in the assembly must be repositioned according to how the parts are assembled. This is illustrated in Figure 4.20 using the slider-crank example.

These two common issues are discussed in this section. We discuss how to extract the kinematic information from mating constraints, construct a kinematic model, and carry out kinematic analysis that determines the location and orientation of individual parts in the assembly. There are several methods proposed for converting the mating constraints to kinematic joints. These include a mating relation-based method (e.g., Kim and Lee, 1989; Kim and Wu, 1990) and a contact condition-based method (e.g., Sinha et al., 2002). There are also many methods developed for kinematic analysis of mechanisms (Dawari and Sen, 2007), including approaches based on configuration space (e.g., Joskowicz, 1990; Lozano-Pérez, 1983; Kim et al., 2003), screw theory (e.g., Adams et al., 1999), ports (e.g., Singh and Bettig, 2004), and features (e.g., Eng et al., 1999).

Before getting into the discussion, a few basic terminologies are mentioned. An assembly may be thought of as a set of rigid bodies connected by joints. A rigid body can be a single part or a

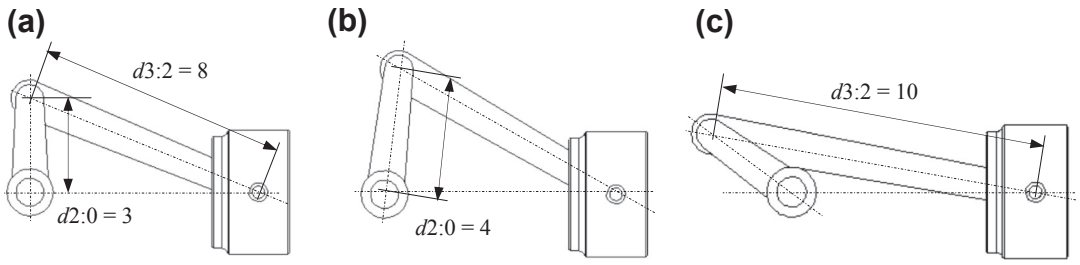


FIGURE 4.19

Design changes in the slider-crank assembly. (a) Dimensions $d2{:}0$ and $d3{:}2$, (b) $d2{:}0$ changed to 4, and (c) $d3{:}2$ changed to 10. All assume a stationary slider.

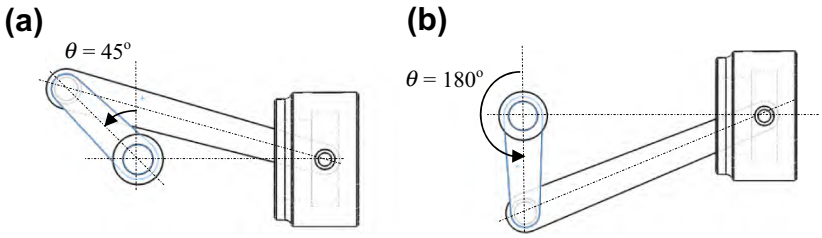


FIGURE 4.20

Crank rotated in the slider-crank assembly. (a) $\theta = 45°$, and (b) $\theta = 180°$.

subassembly, in which no relative motion is allowed between parts. These bodies are called links. An assembly of links and joints creates a kinematic chain, in which links are interconnected in a way to provide a desired output motion in response to an input motion. A mechanism is a kinematic chain in which at least one link has been grounded or attached to the frame of reference.

In this section, we first introduce the method proposed by Kim and Lee (1989) that maps mating constraints to kinematic joints, in which the joint information is automatically extracted from the mating relations for each link. Then, in Section 4.4.2, we discuss the Denavit–Hartenberg (D–H) representation, which is commonly employed to represent kinematic models in robotics applications. We then discuss, in Section 4.4.3, how to construct joint coordinate systems using mating constraints from the CAD assembly to construct a kinematic model. We discuss both open- and closed-loop systems.

4.4.1 MAPPING MATING CONSTRAINTS TO KINEMATIC JOINTS

As discussed in Section 4.2.2, joint constraints impose certain restrictions on the way the components can be assembled, and also on the way they move relative to one another. Each of the joint constraints is related to the rigid-body motion of a mating part and has DOFs associated with it.

As discussed in Section 4.3.2, counting the number of IPVs and types of intersections of the IMG, we are able to determine the remaining DOFs for a pair of mating parts. More precisely, one IPV allows one rotational DOF, and two or more IPVs eliminate all rotational DOFs, as summarized in Table 4.7. Moreover, as shown in Table 4.8, a face IMG allows two translational DOFs (Figure 4.18(a)), a line IMG allows one translational DOF (Figure 4.18(b)), and a point IMG allows no translational movement (Figure 4.18(c)).

Also, by reviewing the description of kinematic joints discussed in Section 4.2.2, an example of DOF analysis in Figure 4.18(b) shows a prismatic joint before and after applying two coincident-aligned constraints. The base and mating parts of the prismatic joint take two planar faces, respectively, as IMGs. Each component has two IPVs and a line IMG; therefore, the joint has zero rotational DOFs and one translational DOF after assembly.

In addition to the case of Figure 4.18(b), in which two coincident-aligned constraints are applied, cases such as two coincident-mates, one mate and one align, and one concentric and one mate (or align), as shown in Figure 4.21(a), map to a prismatic joint.

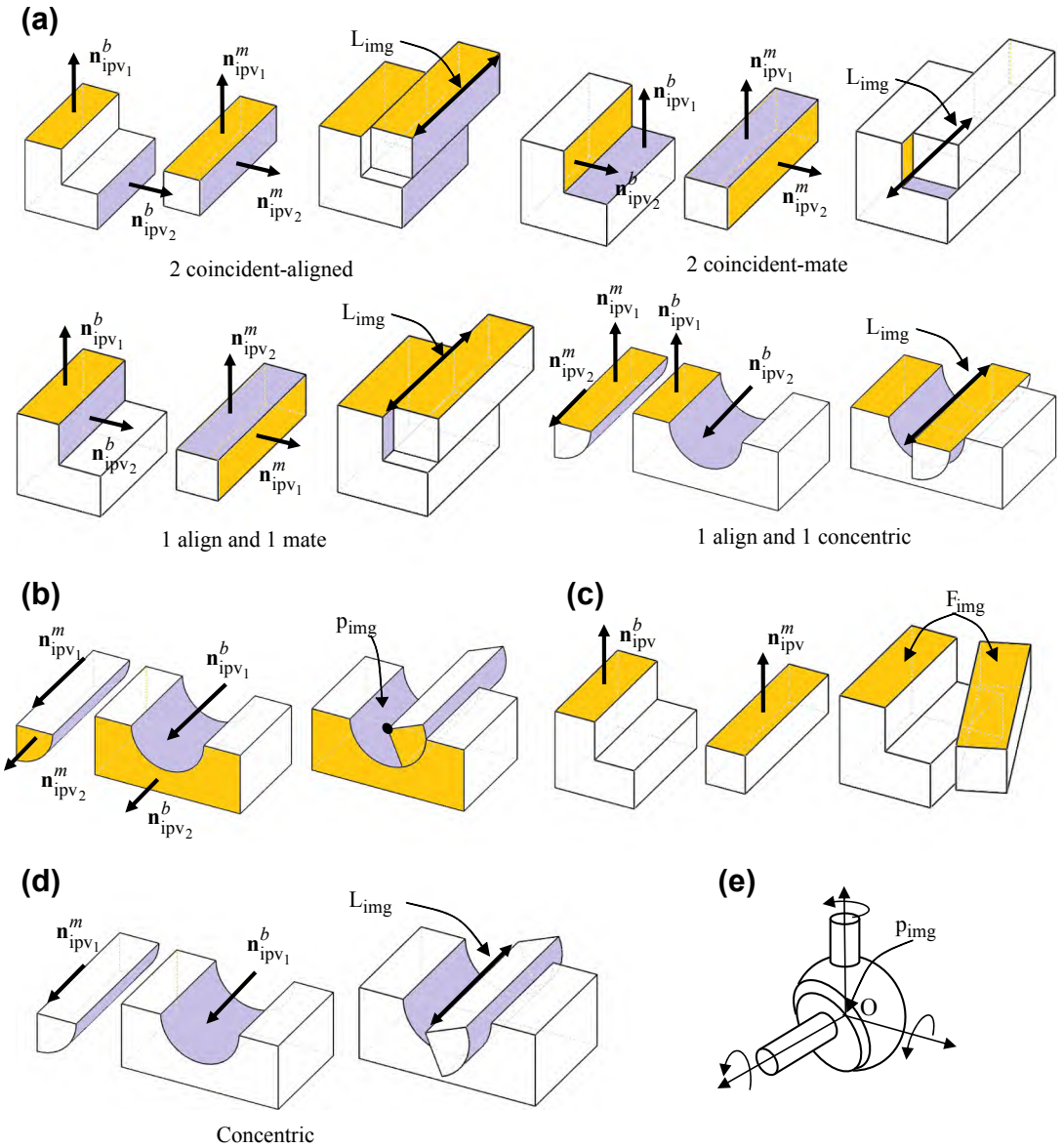


FIGURE 4.21

Illustration of mapping between mating constraints and kinematic joints. (a) Prismatic, (b) revolute, (c) planar, (d) cylindrical, and (e) spherical joints.

Table 4.9 The Mapping between Mating Constraints and Kinematic Joints

Joint	DOF	Number of IPVs	IMGs	Mating Constraints
Prismatic	T1R0	2	Line	Two coincident-mates, two coincident-aligned, coincident-mate and coincident-aligned, or coincident-mate (or align) and concentric
Revolute	T0R1	1	Point	Coincident-mate and concentric, or coincident-aligned and concentric
Planar	T2R1	1	Plane	Coincident-mate, or coincident-aligned
Cylindrical	T1R1	1	Line	Concentric
Spherical	T0R3	0	Point	Point coincident

The example shown in Figure 4.18(c) presents a revolute joint before and after applying one concentric and one coincident-aligned (or coincident-mate) constraints. Each component has one IPV. The point IMG is determined by intersecting the axis (concentric) and the mate plane (coincident-aligned or coincident-mate), as illustrated in Figure 4.21(b). As a result, the mating part is allowed to rotate along the axis (or IPV), resulting in a revolute joint.

With the discussion above, not only the number of DOFs can be determined by counting the number of IPV and checking the type of IMG between two mating parts, but also the type of joint between the two components can be determined. The mapping between mating constraints and kinematic joints is provided in Table 4.9. Figures 4.21(c–e) illustrate the mapping between mating constraints and kinematic joints of planar, cylindrical, and spherical joints, respectively.

4.4.2 D–H REPRESENTATION

After converting mating constraints to kinematic joints, the next step is to construct a kinematic model mathematically. We discuss a modeling approach that is commonly employed in robotics applications.

In robotics applications, most joints are associated with one actuator, either translational or rotational. Therefore, it is commonly assumed that all joints have only a single degree of freedom (Craig, 1989). Note that the assumption does not involve any real loss of generality because joints with multiple DOFs, such as a spherical joint (three rotational DOFs), can always be thought of as a succession of single DOF joints with zero-length links in between. This point is further illustrated in Example 4.4.

With the assumption that each joint has a single DOF, the action of each joint can be described by a single real number—that is, the angle of rotation in the case of a revolute joint or the displacement in the case of a prismatic joint. The objective of the kinematic analysis is to determine the cumulative effects of the entire set of joint variables.

An open-loop mechanism with n joints has $n + 1$ links, as illustrated in Figure 4.22(a) schematically, because each joint connects two links. We number the joints from 1 to n, and we number the links from 0 to n, starting from the ground link 0. By this convention, joint i connects link $i - 1$ to link i. We consider the location of joint i to be fixed with respect to link $i - 1$. For example, Joint 2 is fixed to Link 1 in Figure 4.22(a). When joint i is actuated, link i moves. Again, link 0 (the first link) is fixed and does not move when the joints are actuated.

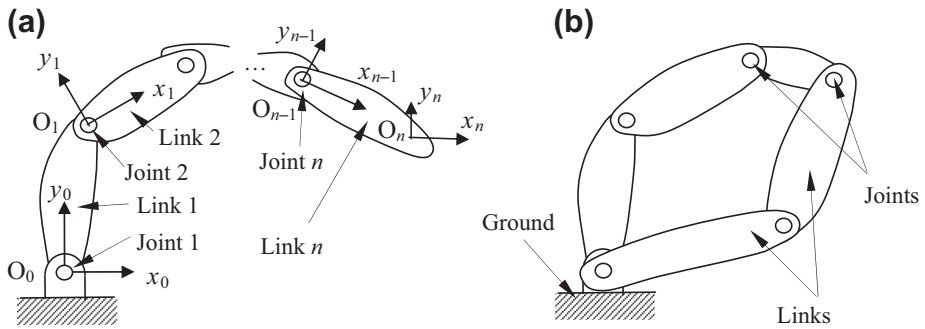


FIGURE 4.22

Schematic representation of kinematic mechanism. (a) Open-loop and (b) closed-loop.

Note that in general link n is not connected back to the base link 0, which is called open-loop. Figure 4.22(b) shows a closed-loop system, in which link n connects back to link 0 (or any link between 0 and $n - 2$).

With the ith joint, we associate a joint variable, denoted by q_i. In the case of a revolute joint, q_i is the angle of rotation; in the case of a prismatic joint, q_i is the joint displacement.

To construct a kinematic model, we rigidly attach a coordinate system to each link at the joint. In particular, we attach x_i–y_i–z_i to link i with an origin O_i at joint $i + 1$. We call this coordinate system C_i, defined by its origin O_i with three axes x_i–y_i–z_i. This means that whatever motion the joint imposes, the location of each point on link i is constant when expressed in the ith coordinate frame. Furthermore, when joint i is actuated, link i and its attached frame C_i experience a resulting motion. The frame C_0, which is attached to the ground link, is referred to as the inertial frame. Note that O_n of link n (or end link), in which the axes x_n–y_n–z_n of the coordinate system C_n are attached, is usually located at a point of interest in design. In robotics applications, C_n is called the end-effector.

The transformation matrix similar to that of Section 4.3 can be employed to express the location and orientation of individual links. For example, matrix $\mathbf{T}_i^{i-1}$ defines the location and orientation of link i with respect to link $i - 1$ or relating coordinate system of C_i with respect to C_{i-1}. More specifically, the matrix transforms a given vector $\mathbf{v}_i = [v_{ix}, v_{iy}, v_{iz}]^T$ in the ith link back to the coordinate system of the $(i - 1)$th link $\mathbf{v}_{i-1} = [v_{i-1x}, v_{i-1y}, v_{i-1z}]^T$ in a homogeneous coordinate system:

$$\begin{bmatrix} \mathbf{v}_{i-1} \\ 1 \end{bmatrix} = \mathbf{T}_i^{i-1} \begin{bmatrix} \mathbf{v}_i \\ 1 \end{bmatrix} \tag{4.12a}$$

in which $\mathbf{T}_i^{i-1}$ is a homogeneous transformation matrix like that of Section 4.3. The assumption that all joints are either revolute or prismatic implies that $\mathbf{T}_i^{i-1}$ is a function of only a single joint variable, namely q_i; that is,

$$\mathbf{T}_i^{i-1} = \mathbf{T}_i^{i-1}(q_i) \tag{4.12b}$$

in which

$$\mathbf{T}_i^{i-1} = \begin{bmatrix} \mathbf{R}_i^{i-1} & \mathbf{L}_i^{i-1} \\ 0 & 1 \end{bmatrix}. \tag{4.12c}$$

Now the homogeneous transformation matrix that expresses the location and orientation of C_j with respect to C_i is denoted by $\mathbf{T}_j^i$, which can be written as

$$\mathbf{T}_j^i = \mathbf{T}_{i+1}^i \mathbf{T}_{i+2}^{i+1} \ldots \mathbf{T}_{j-1}^{j-2} \mathbf{T}_j^{j-1}, \text{ assuming } i < j. \tag{4.13}$$

Note that the following equations are valid because the homogeneous transformation matrices are orthogonal:

$$\mathbf{T}_j^i = \mathbf{I}, \text{ if } i = j, \text{ and} \tag{4.14a}$$

$$\mathbf{T}_j^i = \left(\mathbf{T}_i^j\right)^{-1}. \tag{4.14b}$$

By plugging Eq. 4.12c into Eq. 4.13, we have

$$\mathbf{T}_j^i = \begin{bmatrix} \mathbf{R}_j^i & \mathbf{L}_j^i \\ 0 & 1 \end{bmatrix} \tag{4.15a}$$

where

$$\mathbf{R}_j^i = \mathbf{R}_{i+1}^i \mathbf{R}_{i+2}^{i+1} \ldots \mathbf{R}_j^{j-1} \tag{4.15b}$$

which represents the orientation of the coordinate system C_j relative to coordinate system C_i, and

$$\mathbf{L}_j^i = \mathbf{L}_{j-1}^i + \mathbf{R}_{j-1}^i \mathbf{L}_j^{j-1} \tag{4.15c}$$

denoting the location of the coordinate system C_j relative to coordinate system C_i.

By the manner in which we have rigidly attached the various coordinate systems to the corresponding links, it follows that the position of any point on the end link (Link n), when expressed in coordinate system C_n, is a constant independent of the configuration of the mechanism. Then the location and orientation of the end link in the inertial frame are given by

$$\mathbf{T}_n^0 = \mathbf{T}_1^0(q_1)\mathbf{T}_2^1(q_2)\ldots\mathbf{T}_n^{n-1}(q_n). \tag{4.16a}$$

Note that in general link n is not connected back to the base link 0; therefore, Eq. 4.16 represents an open-loop kinematic system. For a closed-loop system, link n usually connects back to link 0. In this case, Eq. 4.16a becomes

$$\mathbf{I} = \mathbf{T}_1^0(q_1)\mathbf{T}_2^1(q_2)\ldots\mathbf{T}_n^{n-1}(q_n)\mathbf{T}_0^n(q_0) \tag{4.16b}$$

in which the matrix $\mathbf{T}_0^n = (\mathbf{T}_n^0)^{-1}$ is multiplied from the right on both sides of Eq. 4.16a.

It is possible to simplify the transformation matrices by introducing conventions to represent a joint mathematically. In robotics applications, a commonly used convention for selecting frames of reference is the D–H convention (Denavit and Hartenberg, 1955; Hartenberg and Denavit, 1965). Following this convention, a considerable amount of streamlining and simplification in the mathematical

representation of the kinematic model can be achieved. In this convention, each homogeneous transformation matrix $\mathbf{T}_i^{i-1}$ is represented as a product of four basic transformation matrices; that is,

$$\mathbf{T}_i^{i-1} = \mathbf{R}_z(\theta_i)\mathbf{T}_z(d_i)\mathbf{T}_x(a_i)\mathbf{R}_x(\alpha_i) \tag{4.17a}$$

where

$$\mathbf{R}_z(\theta_i) = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.17b}$$

$$\mathbf{T}_z(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.17c}$$

$$\mathbf{T}_x(a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.17d}$$

$$\mathbf{R}_x(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.17e}$$

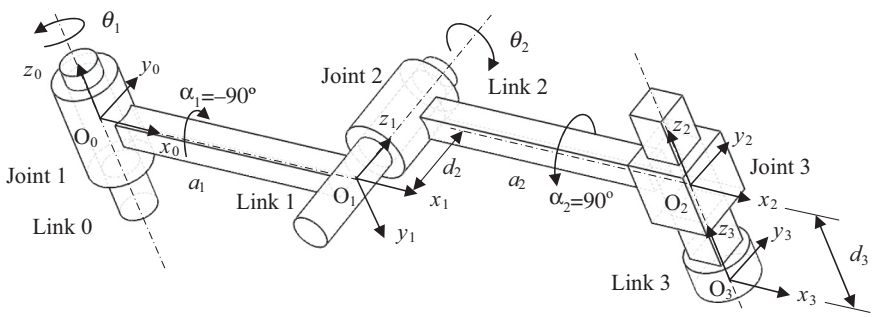


FIGURE 4.23

Illustration of parameters: joint angle, link offset, link length, and link twist.

Note that the four parameters θ_i, d_i, a_i, and α_i, illustrated in Figure 4.23, are associated with link i and joint i, and are generally named joint angle (θ_i), link offset (d_i), link length (a_i), and link twist (α_i), respectively. Also, in Eqs 4.17b and e, the short-hand notations, such as $c\theta_i = \cos\theta_i$, $s\alpha_i = \sin\alpha_i$, are employed.

As an example, the system shown in Figure 4.23 consists of four links (0, 1, 2, and 3) and three joints. Note that Joints 1 and 2 are revolute, and Joint 3 is a prismatic joint. θ_1 and θ_2 are the joint angles of the revolute Joints 1 and 2, respectively. a_1 and a_2 are the link lengths of Links 1 and 2, respectively. α_1 and α_2 are the link twists of Links 1 and 2, respectively. d_2 is the offset between coordinate systems C_1 and C_2 (not necessarily a joint offset in this case). d_3 is the joint offset of the prismatic joint (Joint 3). Note that in this system, link lengths a_1 and a_2, offset d_2, and link twists α_1 and α_2 are constant.

Plugging Eqs 4.17b–e into Eq. 4.17a, we have

$$\mathbf{T}_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_i^{i-1} & \mathbf{L}_i^{i-1} \\ 0 & 1 \end{bmatrix} \tag{4.18a}$$

where

$$\mathbf{R}_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i \\ 0 & s\alpha_i & c\alpha_i \end{bmatrix} \tag{4.18b}$$

and

$$\mathbf{L}_i^{i-1} = \begin{bmatrix} a_i c\theta_i \\ a_i s\theta_i \\ d_i \end{bmatrix} \tag{4.18c}$$

in which $\mathbf{R}_i^{i-1}$ and $\mathbf{L}_i^{i-1}$ are the rotation and translation matrices, respectively. In other words, $\mathbf{R}_i^{i-1}$ and $\mathbf{L}_i^{i-1}$ orient and locate Link i with respect to Link $i-1$.

Because the matrix $\mathbf{T}_i^{i-1}(q_i)$ is a function of a single variable q_i, as defined in Eq. 4.11, three of the four parameters in each individual transformation matrice are constant. Only one parameter is allowed to vary. It is apparent that for a revolute joint, joint angle θ_i is the variable; for a prismatic joint, link offset d_i is the only variable. For the system shown in Figure 4.23, joint angles θ_1 and θ_2 and the joint offset d_3 are variables.

One important note to make is that although the choices of coordinate systems are not unique, they have to be chosen carefully. If the coordinate systems chosen satisfy the following two conditions, then there exist unique numbers θ_i, d_i, a_i, and α_i such that Eq. 4.17a can be determined (Spoong et al., 2005):

(1) The axis x_i is perpendicular to the axis z_{i-1}, and
(2) The axis x_i intersects the axis z_{i-1}.

Note that the choice of the origin of the coordinate system is less restrictive in general.

Table 4.10 List of Link Parameters for the System Shown in Figure 4.23

Link	θ_i	d_i	a_i	α_i
1	θ_1	0	a_1	$\alpha_1 = -90^\circ$
2	θ_2	d_2	a_2	$\alpha_2 = 90^\circ$
3	0	d_3	0	0

The coordinate systems defined for the links of the example shown in Figure 4.23 satisfy the conditions. A more important convention in choosing coordinate systems is to assign z_i to be the axis of actuation for joint $i + 1$. For example, the axis z_1 in Figure 4.23 is assigned at the actuation direction (axis of rotation) of the revolute joint, Joint 2.

To completely define a coordinate system, we need the origin and axis x; then, the remaining y-axis can be determined by the right-hand rule. We will use examples to illustrate the construction of coordinate systems momentarily. For the time being, we assume coordinate systems for a given kinematic model have been created, and we discuss the measurement of the four parameters: joint angle (θ_i), link offset (d_i), link length (a_i), and link twist (α_i).

- Joint angle θ_i is the required rotation of x_{i-1}-axis about the z_{i-1}-axis to become parallel to the x_i-axis. For example, θ_1 in Figure 4.23 is the rotation angle of x_0-axis about the z_0-axis to become parallel to x_1-axis.
- Joint distance d_i is the distance between the x_{i-1} and x_i axes along the z_{i-1}-axis. Joint distance is also called link offset. For example, d_3 in Figure 4.23 is the distance between x_2 and x_3 about the z_2-axis.
- Link twist α_i is the required rotation of the z_{i-1}-axis about the x_i-axis to become parallel to the z_i-axis. For example in Figure 4.23, α_1 is the required rotation of the z_0-axis about the x_1-axis to become parallel to the z_1-axis, which is -90° in this case.
- Link length a_i is the distance between z_{i-1} and z_i axes along the x_i-axis. Note that a_i is the kinematic length of link i. For example, in Figure 4.23, a_1 is the distance between z_0 and z_1 axes along the x_1-axis.

Apparently, the location of the origin of a coordinate system could affect link offset d_i and link length a_i.

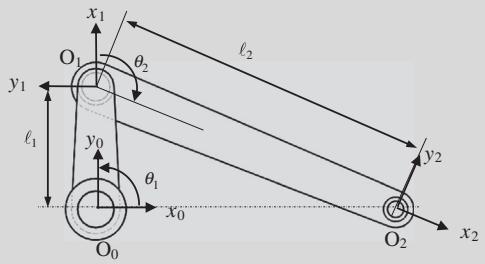
Once the link parameters are identified, a table that lists link parameters for the system can be created. For example, the table for the system shown in Figure 4.23 can be created as in Table 4.10, with which transformation matrices can be written readily using Eq. 4.18a.

Note that link parameters a_1, a_2, α_1, α_2, and d_2 in Table 4.10 are constant.

The following example provides further illustration on the calculation of the transformation matrices. In Examples 4.3 and 4.4, we assigned coordinate systems that satisfy the two conditions mentioned above. In later examples, we illustrate the rules of specifying origin and x-axis of individual coordinate systems for both open- and closed-loop assemblies.

EXAMPLE 4.3

Consider a planar two-bar system shown below, which consists of two revolute joints and three links.



Note that joint axes z_i, $i = 0, 2$, are normal to the page, and all coordinate systems assigned satisfy the two conditions stated above. The base frame C_0 is fixed to Link 0 as shown. Calculate the transformation matrix for Link 2 with respect to Link 0; that is, $\mathbf{T}_2^0$.

Solutions

We first create a table that lists link parameters for the system as shown below.

Link	θ_i	d_i	a_i	α_i
1	θ_1	0	$a_1 = \ell_1$	0
2	θ_2	0	$a_2 = \ell_2$	0

From Eq. 4.18a, we have

$$\mathbf{T}_1^0 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & \ell_1 c\theta_1 \\ s\theta_1 & c\theta_1 & 0 & \ell_1 s\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and } \mathbf{T}_2^1 = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & \ell_2 c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & \ell_2 s\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Therefore, from Eq. 4.16a, we have

$$\mathbf{T}_2^0 = \mathbf{T}_1^0 \mathbf{T}_2^1 = \begin{bmatrix} c\theta_{12} & -s\theta_{12} & 0 & \ell_1 c\theta_1 + \ell_2 c\theta_{12} \\ s\theta_{12} & c\theta_{12} & 0 & \ell_1 s\theta_1 + \ell_2 s\theta_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $c\theta_{12} = \cos(\theta_1 + \theta_2)$, and $s\theta_{12} = \sin(\theta_1 + \theta_2)$.

Note that the first two entries of the last column of $\mathbf{T}_2^0$ are the x and y components of the origin O_2 referring to the base frame; that is,

$$O_{2x} = \ell_1 \cos \theta_1 + \ell_2 \cos (\theta_1 + \theta_2)$$

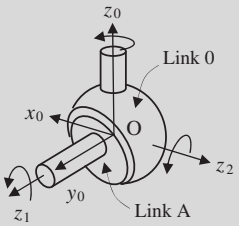
$$O_{2y} = \ell_1 \sin \theta_1 + \ell_2 \sin (\theta_1 + \theta_2).$$

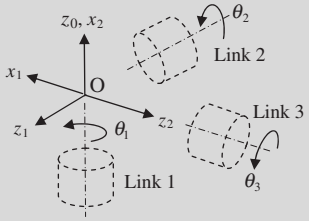
The rotation part of $\mathbf{T}_2^0$ defines the orientation of the coordinate system C_2 relative to the base frame.

We mentioned earlier that a joint of multiple DOFs, such as a spherical joint of three rotational DOFs, can always be thought of as a succession of single DOF joints with zero-length links in between. We derive the transformation matrix for a spherical joint in the following example.

EXAMPLE 4.4

A spherical joint shown below (left) is defined by three joint axes, z_0, z_1, and z_2, which intersect at point O. Physically, the spherical joint connects Link A to Link 0 (base part). This spherical joint can be thought of as a succession of revolute joints with zero-length links in between, as illustrated in the figure below (right).





Derive a transformation matrix for the spherical joint.

Solutions

We first define the coordinate systems for the links, as shown above (right), satisfying the two conditions. Based on the coordinate systems, we have three joint angles, θ_1, θ_2, and θ_3, and two twist angles, α_1 and α_2, which are nonzero. Note that $\alpha_1 = -90^\circ$ and $\alpha_2 = 90^\circ$, according to the way that angles are measured as stated earlier.

We create a table that lists link parameters as below.

Link	θ_i	d_i	a_i	α_i
1	θ_1	0	0	-90°
2	θ_2	0	0	90°
3	θ_3	0	0	0

From Eq. 4.18a, we have

$$\mathbf{T}_1^0 = \begin{bmatrix} c\theta_1 & 0 & -s\theta_1 & 0 \\ s\theta_1 & 0 & c\theta_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_2^1 = \begin{bmatrix} c\theta_2 & 0 & s\theta_2 & 0 \\ s\theta_2 & 0 & -c\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{and } \mathbf{T}_3^2 = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & 0 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, from Eq. 4.16a, we have

$$\mathbf{T}_3^0 = \mathbf{T}_1^0\mathbf{T}_2^1\mathbf{T}_3^2 = \begin{bmatrix} c\theta_1c\theta_2c\theta_3 - s\theta_1s\theta_3 & -c\theta_1c\theta_2s\theta_3 - s\theta_1c\theta_3 & c\theta_1s\theta_2 & 0 \\ s\theta_1c\theta_2c\theta_3 + c\theta_1s\theta_3 & -s\theta_1c\theta_2s\theta_3 + c\theta_1c\theta_3 & s\theta_1s\theta_2 & 0 \\ -s\theta_2c\theta_3 & s\theta_2s\theta_3 & c\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Note that $\mathbf{T}_3^0 = \mathbf{T}_A^0$, which transforms the rotation of Link A back to the base link 0.

So far in the examples we discussed, all coordinate systems assigned satisfy the two important conditions. For an assembly model created by using kinematic joints in CAD, these coordinate systems can be created systematically. We use the slider-crank example shown in Figure 4.11 to illustrate the details. We first discuss open-loop system, in which we remove the prismatic joint between the slider and the ground. Then we discuss closed-loop system by resuming the prismatic joint.

4.4.2.1 Open-Loop System

First, as discussed earlier, the z-axis of a joint aligns with the joint actuation direction. For a revolute joint, the z-axis aligns with the axis of rotation. Hence, the z-axes for all the three revolute joints are determined and illustrated in Figure 4.24(a). Note that the positive direction of the z-axis is determined by Pro/ENGINEER internally, depending on the orientation of the datum axes selected for the individual joints. Users may flip the positive direction of a joint. In this example, the joint directions were adjusted to be pointing in the same direction as shown.

Next, the origin of individual coordinate systems associated with joints can be assigned at the datum points that were employed for defining the joints or at the intersection of joint axis and the

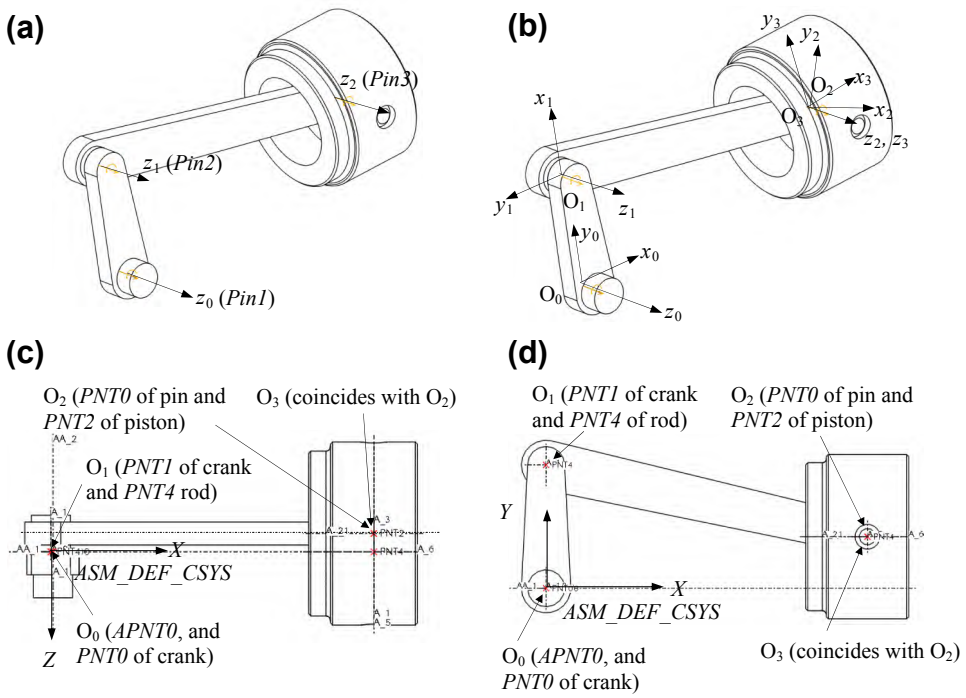


FIGURE 4.24

Determining the coordinate systems: (a) Z-axes, (b) coordinate systems, (c) origins of the coordinate systems (top view), and (d) origins of the coordinate systems (front view).

mating faces. For example, for the first pin joint Pin1, the origin O_0 is located where datum points *PNT0* (crank) and *APNT0* coincide, as illustrated in Figures 4.24(c) and (d). Similarly, O_1 (*PNT1* of crank and *PNT4* of rod), and O_2 (*PNT0* of pin and *PNT2* of piston) can be located. For O_3, because there is no datum point involved, O_3 can be located at any point along z_3; for example, locating O_3 to coincide with O_2, as shown in Figure 4.24(b).

Now, we set the base coordinate system C_0. Because the origin O_0 and axis z_0 are determined, all we need is to determine axis x_0; axis y_0 can then be determined by the right-hand rule. Choosing axis x_0 is arbitrary for an open-loop system. For convenience, we may choose it to align with that of the global coordinate system (*ASM_DEF_CSYS* shown in Figures 4.24(c) and (d)) as long as the x-axis of the global coordinate system is not parallel to the axis z_0. The x_0 axis is determined as shown in Figure 4.24(b); hence, the y_0 axis and the coordinate system C_0 are determined.

After setting the base coordinate system C_0, we are now ready to assign x-axes for the remaining coordinate systems. There are three possible cases we considered. We discuss only Case A for the time being, which is relevant to the current open-loop example. If the axes z_i and z_{i-1} are parallel (such as z_1 and z_0, and z_2 and z_1 shown in Figure 4.24(a)), the axis x_i is chosen to be directed from O_i toward z_{i-1}, or as the opposite of this vector. In this example, we choose the latter: x_1 is chosen to be directed from O_1 toward the z_0-axis, but in the opposite direction, as shown in Figure 4.24(b). Similarly, x_2 can be determined the same way. This is Case A, which is all we need for this open-loop example. More cases are discussed in the next example, closed-loop.

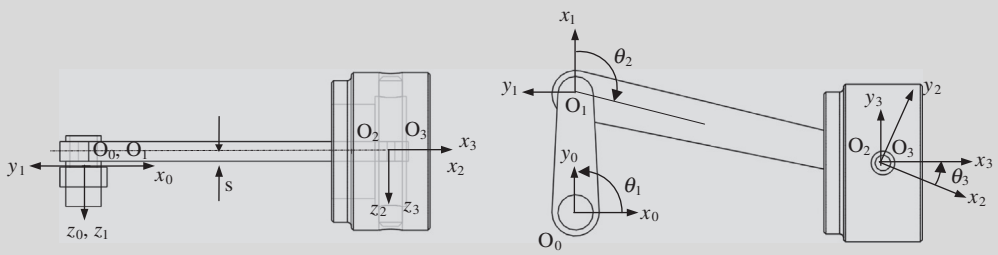
The coordinate system C_3 is added with its origin coinciding with that of C_2 and rotates θ_3 angle along the axis z_2, so that x_3-axis is parallel to x_0 for the time being, as shown in Figure 4.24(b). Again, the coordinate system C_3 is called the end-effector in robotics applications.

Note that the approach of determining coordinate systems discussed above is systematic and general, which satisfy the two conditions mentioned above and can be implemented into a computer. Once the individual coordinate systems are determined, the transformation matrices of the kinematic model can be created following the same approach discussed earlier.

We illustrate the calculation of the transformation matrices for the open-loop system in the following example.

EXAMPLE 4.5

Calculate the transformation matrices for the slider-crank mechanism shown in Figure 4.24(b). The top and front views of the mechanism are sketched below with coordinate systems shown.



Continued

EXAMPLE 4.5—cont'd

Solutions

We first create a table that lists link parameters for the system as below.

Link	θ_i	d_i	a_i	α_i
Crank (1)	θ_1	0	$a_1 = \ell_1$	0
Rod (2)	θ_2	$d_2 = -s$	$a_2 = \ell_2$	0
Slider (3)	θ^3	0	0	0

Note that although the rod does not involve any prismatic joint, its origin O_2 is offset a $-s$ amount (constant) from O_1 along the z_1-axis.

From Eq. 4.18a, we have

$$\mathbf{T}_1^0 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & \ell_1 c\theta_1 \\ s\theta_1 & c\theta_1 & 0 & \ell_1 s\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_2^1 = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & \ell_2 c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & \ell_2 s\theta_2 \\ 0 & 0 & 1 & -s \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{and } \mathbf{T}_3^2 = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & 0 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Therefore, from Eq. 4.16a, we have

$$\mathbf{T}_3^0 = \mathbf{T}_1^0 \mathbf{T}_2^1 \mathbf{T}_3^2 = \begin{bmatrix} c\theta_{123} & -s\theta_{123} & 0 & \ell_1 c\theta_1 + \ell_2 c\theta_{12} \\ s\theta_{123} & c\theta_{123} & 0 & \ell_1 s\theta_1 + \ell_2 s\theta_{12} \\ 0 & 0 & 1 & -s \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $c\theta_{123} = \cos(\theta_1 + \theta_2 + \theta_3)$, and $s\theta_{123} = \sin(\theta_1 + \theta_2 + \theta_3)$.

Note that the first three entries of the last column of $\mathbf{T}_3^0$ are the x-, y-, and z-components of the origin O_3 referring to the base frame; that is,

$$O_{3x} = \ell_1 \cos \theta_1 + \ell_2 \cos (\theta_1 + \theta_2)$$

$$O_{3y} = \ell_1 \sin \theta_1 + \ell_2 \sin (\theta_1 + \theta_2)$$

$$O_{3z} = -s.$$

In addition, the rotation part of $\mathbf{T}_3^0$ defines the orientation of the coordinate system C_3 relative to the base frame C_0.

If we add a parallel mating constraint (called the align-oriented constraint in Pro/ENGINEER) between the horizontal plane (DTM3) of the piston and the datum plane *ASM_TOP*, as shown in Figure 4.25, the piston is allowed to translate on the x_0–y_0 plane, but it is not allowed to rotate. In this case, the rotation part of the of $\mathbf{T}_3^0$ becomes an identify matrix; that is,

$$\cos(\theta_1 + \theta_2 + \theta_3) = 1, \text{and } \sin(\theta_1 + \theta_2 + \theta_3) = 0.$$

Hence

$$\theta_1 + \theta_2 + \theta_3 = 0 \text{ (or } 180^\circ).$$

In this case, the system is no longer an open-loop.

In the following example, we illustrate the usage of the transformation matrices, in particular, to calculate the joint parameters in order to determine the configuration of the assembly.

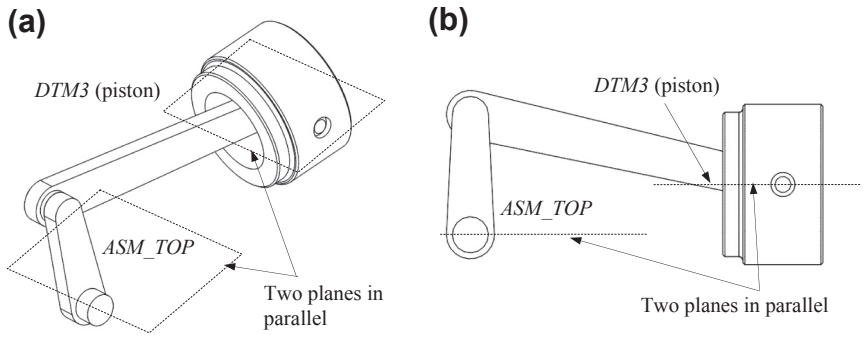
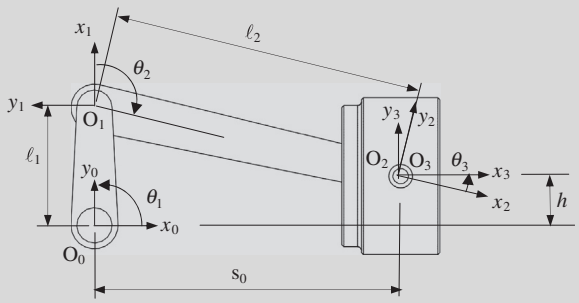


FIGURE 4.25

Parallel mating constraint added between two planes: (a) iso-view, and (b) front view.

EXAMPLE 4.6

We continue with Example 4.5 and assume that $\theta_1 = 90°$, $\ell_1 = 3$, and $\ell_2 = 8$. The vertical distance between the piston and the inertia frame is $h = 1$, as shown in the figure below. Determine the configuration of the assembly by calculating parameters θ_2, θ_3, and the distance between the piston and the base frame s_0, as shown in the figure below.



Solutions

The vertical position of the piston is given as $h = 1$, then

$$O_{3y} = \ell_1 \sin\theta_1 + \ell_2 \sin\left(\theta_1 + \theta_2\right) = \ell_1 + \ell_2 \sin\left(90 + \theta_2\right) = h.$$

The angle θ_2 can be solved as

$$\theta_2 = \sin^{-1}\left(\frac{h - \ell_1}{\ell_2}\right) - 90° = \sin^{-1}\left(\frac{1-3}{8}\right) - 90° = -105° \text{or } 105°$$

giving two possible configurations, $\theta_2 = -105°$ shown above, and $\theta_2 = 105°$, where the piston is positioned to the left of the crank (see figure on next page).

Then the x-position of the slider can be found as

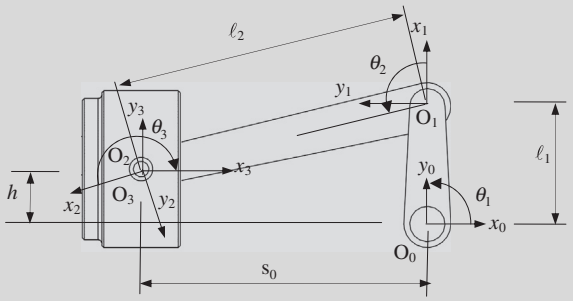
$$s_0 = O_{3x} = \ell_1 \cos\theta_1 + \ell_2 \cos\left(\theta_1 + \theta_2\right) = \ell_2 \cos\left(\theta_1 + \theta_2\right) = 8\cos(90° \pm 105°) = \pm 7.73.$$

Note that $s_0 = 7.73$, indicating the configuration shown above, and the configuration of $s_0 = -7.73$ is shown on next page.

If the parallel mating constraint is present, the angle θ_3 can be calculated as
$\theta_3 = -(\theta_1 + \theta_2) = -(90° \pm 105°) = 15°$ (see figure above) or $-195°$ (see figure on next page).

Continued

EXAMPLE 4.6—cont'd



4.4.2.2 Closed-Loop System

Now, we resume the prismatic joint and discuss the slider-crank mechanism as a closed-loop system.

For a closed-loop system, the x-axis of the base coordinate system C_0 cannot be determined arbitrary in general. For the time being, we assume a closed-loop system, in which the last link n is connected back to the ground link 0. The x_0-axis must be determined as if link 0 is connected to link n—that is, determined by axes z_0 and z_n following the same rule as any other joints as discussed above.

For the slider-crank example, the slider is connected back to the ground via a prismatic joint, and the z-axis aligns with its translational direction. With the z-axes of the three revolute joints shown before, the z-axes for all the four joints are determined and illustrated in Figure 4.26(a). The origins of coordinate systems associated with joints are identical to those of the open-loop example, except for O_3. O_3 is assigned to datum point *PNT4* because the axis *A_6* that defines the translational direction of the prismatic joint passes *PNT4*. Note that in this case, O_3 is offset s from O_2 along z_2-axis.

Next, we set the x-axis for the coordinate systems C_1 and C_2 as before (Case A). However, for coordinate system C_3, the axis z_3 is not in parallel with z_2; instead, they intersect. For cases where z_i intersects z_{i-1} (Case B), x_i is chosen normal to the plane formed by z_i and z_{i-1} (with positive chosen arbitrarily). Hence, x_3 is determined by pointing (for example) upward, and y_3 is also determined by the right-hand rule as shown in Figure 4.26(b).

Now, the slider connects back to the ground, and we must determine the x-axis of the coordinate system C_0. Because axis z_3 intersects z_0, we have Case B. Hence, x_0 is determined pointing (for example) upward, and y_0 is also determined by the right-hand rule as shown in Figure 4.26(b).

Note that in both Cases A and B, z_i and z_{i-1} are coplanar. If z_i and z_{i-1} are not coplanar (Case C), for example, axes z_1 and z_0 shown in Figure 4.23, then there exists a line segment perpendicular to both z_i and z_{i-1} such that it connects both axes and it has a minimum length. The line containing this common normal to z_i and z_{i-1} defines x_i, and the axis y_i is determined to form a right-hand frame.

Note that the approach of determining coordinate systems discussed above is systematic and general, which satisfy the two conditions mentioned above and can be implemented into the

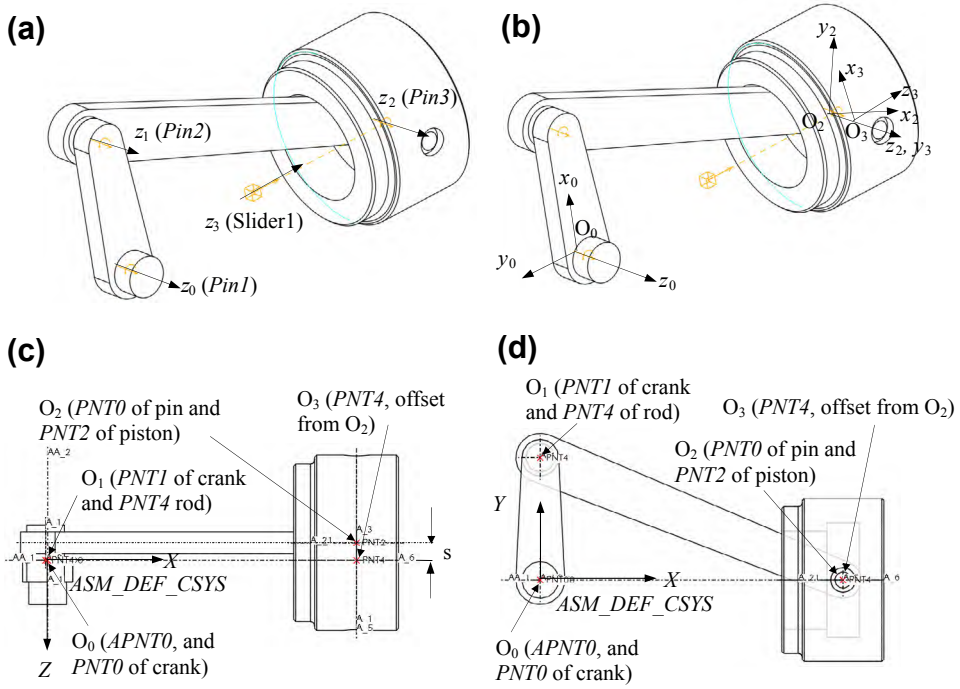


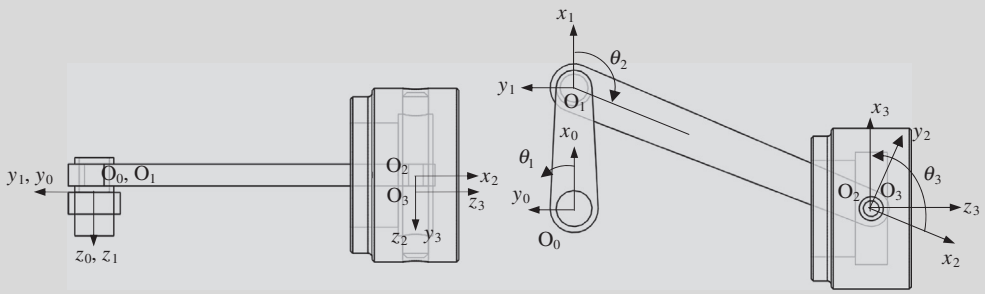
FIGURE 4.26

Determining the coordinate systems: (a) *Z*-axes, (b) coordinate systems, (c) origins of the coordinate systems (top view), (d) and origins of the coordinate systems (front view).

computer. Once the individual coordinate systems are determined, the transformation matrices of the kinematic model can be created following the same approach discussed earlier. Calculation of the transformation matrices for the slider-crank mechanism is illustrated in the following example.

EXAMPLE 4.7

Calculate the transformation matrices for the slider-crank mechanism shown in Figure 4.26(b). The top and front views of the mechanism are sketched below with coordinate systems shown.



Continued

EXAMPLE 4.7—cont'd

Solutions

We first create a table that lists link parameters for the system as below.

Link	θ_i	d_i	a_i	α_i
Crank (1)	θ_1	0	$a_1 = \ell_1$	0
Rod (2)	θ_2	$d_2 = -s$	$a_2 = \ell_2$	0
Slider (3)	θ_3	$d_3 = s$	0	90°
Ground (0)	0	d_0	0	−90°

Note that in this example, θ_1, θ_2, θ_3, and d_0 are variables.

From Eq. 4.18a, we have

$$\mathbf{T}_1^0 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & \ell_1 c\theta_1 \\ s\theta_1 & c\theta_1 & 0 & \ell_1 s\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_2^1 = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & \ell_2 c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & \ell_2 s\theta_2 \\ 0 & 0 & 1 & -s \\ 0 & 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_3^2 = \begin{bmatrix} c\theta_3 & 0 & s\theta_3 & 0 \\ s\theta_3 & 0 & -c\theta_3 & 0 \\ 0 & 1 & 0 & s \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{and}$$

$$\mathbf{T}_3^0 = \begin{bmatrix} c\theta_3 & 0 & s\theta_3 & 0 \\ s\theta_3 & 0 & -c\theta_3 & 0 \\ 0 & 1 & 0 & d_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, from Eq. 4.16b, we have

$$\mathbf{I} = \mathbf{T}_1^0 \mathbf{T}_2^1 \mathbf{T}_3^2 \mathbf{T}_0^3 = \begin{bmatrix} c\theta_{123} & -s\theta_{123} & 0 & \ell_1 c\theta_1 + \ell_2 c\theta_{12} + d_0 s\theta_{123} \\ s\theta_{123} & c\theta_{123} & 0 & \ell_1 s\theta_1 + \ell_2 s\theta_{12} - d_0 c\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Hence,

$$c\theta_{123} = \cos(\theta_1 + \theta_2 + \theta_3) = 1, \text{ and } s\theta_{123} = \sin(\theta_1 + \theta_2 + \theta_3) = 0,$$

$$\ell_1 \cos\theta_1 + \ell_2 \cos(\theta_1 + \theta_2) = 0$$

$$\ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1 + \theta_2) - d_0 = 0.$$

Assuming $\theta_1 = 0°$, $\ell_1 = 3$, and $\ell_1 = 8$; we solve θ_2, θ_3 and d_0 from the above equations as follows:

$$\theta_2 = \cos^{-1}\left(\frac{-\ell_1 \cos\theta_1}{\ell_2}\right) - \theta_1 = \cos^{-1}\left(\frac{-3}{8}\right) - 0 = 112.0° \quad \text{or} \quad -112.0°$$

which again gives two possible configurations, $\theta_2 = -112°$ shown on the previous page (right), and $\theta_2 = 112°$ where the piston is positioned to the left of the crank (see figure on next page).

Now the location of the piston can be found as follows:

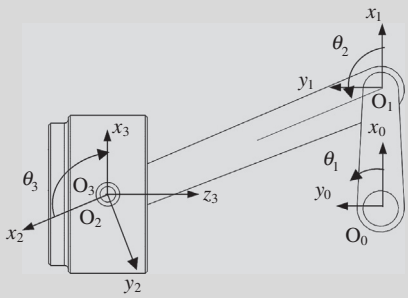
$$d_0 = \ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1 + \theta_2) = \ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1 + \theta_2) = 3(0) + 8 \sin(\pm 112.0) = \pm 7.416.$$

Note that $d_0 = -7.416$, indicating the configuration shown on the previous page (right), and the configuration of $d_0 = 7.416$ is shown on the next page. The sign of the link parameter d_0 is determined by the positive direction of the z_3 axis.

The angle θ_3 can be calculated as

$\theta_3 = -(\theta_1 + \theta_2) = -(\pm 112°) = 112°$ (see figure on previous page) or $-112°$ (figure below).

EXAMPLE 4.7—cont'd



Now, let us go over the scenarios discussed at the beginning of this section. First, we change the dimensions of the crank and connecting rod for the slider-crank mechanism, as shown in Figure 4.19, and determine the location and orientation of the individual parts. We only take one set of link parameter values that result in a configuration with piston on the right to simplify the discussion.

EXAMPLE 4.8

Change the lengths of the crank and rod to 4 and 10, respectively; and calculate the parameters that determine the location and orientation of individual parts. First for Part A, we assume that the angle θ_1 is $\theta_1 = 0$. Then, in Part B, we assume the distance d_0 is a constant $d_0 = 7.416$.

Solutions

For Part A, we assume $\theta_1 = 0^\circ$, $\ell_1 = 4$, and $\ell_2 = 10$; we solve θ_2, θ_3, and d_0 from the above equations as follows:

$$\ell_1 \cos\theta_1 + \ell_2 \cos(\theta_1 + \theta_2) = 0$$

$$\ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1 + \theta_2) - d_0 = 0.$$

Solve for θ_2 from the first equation,

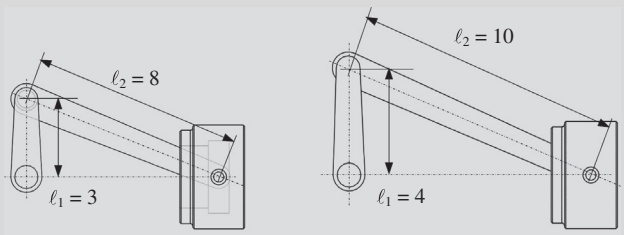
$$\theta_2 = \cos^{-1}\left(\frac{-\ell_1 \cos\theta_1}{\ell_2}\right) - \theta_1 = \cos^{-1}\left(\frac{-4}{10}\right) - 0 = \pm 114^\circ.$$

We take only one value $\theta_2 = -114^\circ$ for discussion, indicating the configuration where the piston is on the right of the crank, as shown below.

With this, we solve for d_0

$$d_0 = \ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1 + \theta_2) = 4(0) + 10 \sin(-114) = -9.165.$$

The figures below, left and right, show the assembly before and after changes, respectively.



Continued

EXAMPLE 4.8—cont'd

Now, for Part B, we assume the distance d_0 is a constant; that is, $d_0 = -7.416$. Note that we add a negative sign to the d_0 value in order to keep it consistent with the given configuration. With $\ell_1 = 4$ and $\ell_2 = 10$, we solve θ_1, θ_2, and θ_3 as follows.

$$\ell_1 \cos\theta_1 + \ell_2 \cos(\theta_1 + \theta_2) = 4 \cos\theta_1 + 10 \cos(\theta_1 + \theta_2) = 0$$

$$\ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1 + \theta_2) - d_0 = 4 \sin\theta_1 + 10 \sin(\theta_1 + \theta_2) + 7.416 = 0.$$

The above nonlinear coupled equations can be solved using, for example, Matlab. For example, the Matlab script shown below (in italic) generates two sets of results.

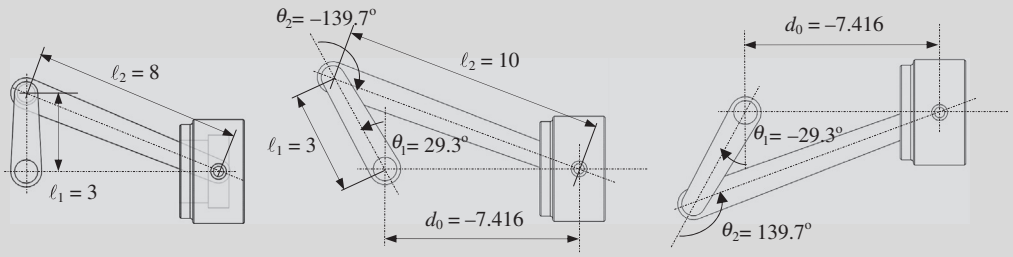
$$[q1, q2] = solve('4^* cos(q1) + 10^* cos(q_1 + q_2) = 0','4^* sin(q_1) + 10^* sin(q_1 + q_2) = -7.416','Real',true).$$

They are:

$$\text{Set 1: } \theta_1 = 0.51078 = 29.3^\circ \text{ and } \theta_2 = -2.4380 = -139.7^\circ$$

$$\text{Set 2: } \theta_1 = 2.6308 = 150.7^\circ (\text{or} -29.3^\circ) \text{ and } \theta_2 = 2.4380 = 139.7^\circ$$

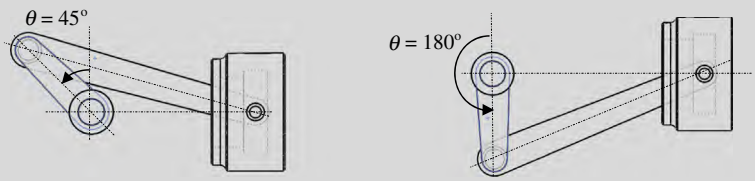
which results in two configurations shown below. The figure on the left shows the mechanism before design changes. The one in the middle indicates the configuration of the solutions of Set 1. The one on the right results from Set 2.



Now, we rotate the crank and reposition the parts in the assembly of the slider-crank example. This is illustrated in the next example. Again, we are only taking one set of link parameter values, which result in the configuration of the piston on the right, to simplify the discussion.

EXAMPLE 4.9

Change the angle θ_1 to 45° (Part A) and then 180° (Part B) as shown below, and calculate the parameters that determine the location and orientation of individual parts.



EXAMPLE 4.9—cont'd

Solutions

We solve θ_2 and d_0 as follows, assuming $\ell_1 = 3$ and $\ell_2 = 8$, and $\theta_1 = 45°$ (Part A).

$$\ell_1 \cos\theta_1 + \ell_2 \cos(\theta_1 + \theta_2) = 0$$

$$\ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1 + \theta_2) - d_0 = 0$$

Solve for θ_2, with $\theta_1 = 45°$,

$$\theta_2 = \cos^{-1}\left(\frac{-\ell_1 \cos\theta_1}{\ell_2}\right) - \theta_1 = \cos^{-1}\left(\frac{-3\cos 45°}{8}\right) - 45° = -150.4° \text{ and } 60.4°$$

which indicates two respective configurations of the mechanism. We take only one value $\theta_2 = -150.4°$ for discussion, indicating the configuration where the piston is on the right of the crank.

Now, we solve for d_0:

$$d_0 = \ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1 + \theta_2) = 3\sin(45) + 8\sin(45 - 150.4) = -5.59$$

which shows the configuration in the figure above (previous page, left).

Now, we solve for θ_2, with $\theta_1 = 180°$ (Part B).

$$\theta_2 = \cos^{-1}\left(\frac{-\ell_1 \cos\theta_1}{\ell_2}\right) - \theta_1 = \cos^{-1}\left(\frac{-3\cos 180°}{8}\right) - 180° = \pm 112°$$

Again, the angles indicate two respective configurations of the mechanism. Like before, we take only one value, $\theta_2 = 112°$, for discussion. Now, we solve for d_0:

$$d_0 = \ell_1 \sin\theta_1 + \ell_2 \sin(\theta_1 + \theta_2) = 3\sin(180) + 8\sin(180 + 112) = -7.416.$$

4.4.3 CONSTRUCTING THE JOINT COORDINATE SYSTEMS

The discussion presented in Section 4.4.2 assumes that the kinematic joints have been well defined in the assembly. This assumption is true if designers use CAD software, such as Pro/ENGINEER, and define kinematic joints using geometric entities such as datum axis, datum points, and so on. These datum entities can be used to determine the z-axis and origin of individual coordinate systems, construct transformation matrices, and solve for the location and orientation for individual links.

However, in most CAD systems, designers use mating constraints, instead of kinematic joints, to create assemblies. How do we construct transformation matrices and solve these equations for the location and orientation of individual parts? The missing link is the z-axis and the origin of the coordinate systems. If the information can be extracted from mating constraints, the same approach discussed in Section 4.4.2 can readily take over the remaining steps in positioning individual components in the assembly. Can the required information be extracted from mating constraints? The answer is yes. In this subsection, we introduce a method proposed by Kim et al. (2001) and Kim et al. (2004).

In Section 4.3, we learned that the type of joints embedded in the CAD assembly can be determined by counting the number of IPVs and the type of IMG revealed in the mating constraints between the two mating parts. In this section, we discuss how to extract information from mating constraints and joint types in order to determine the z-axis and origin of the coordinate systems.

First, for a prismatic joint (Figure 4.27(a)) that was formed by, for example, two coincident-aligned constraints, the direction of the joint axis is determined by the direction in which the joint moves. This joint axis can be determined by

$$\mathbf{n}_t = \mathbf{n}^{mr}_{\text{ipv}_1} \times \mathbf{n}^{mr}_{\text{ipv}_2} \quad \text{or} \quad \mathbf{n}_t = \mathbf{n}^{b}_{\text{ipv}_1} \times \mathbf{n}^{b}_{\text{ipv}_2}. \tag{4.19}$$

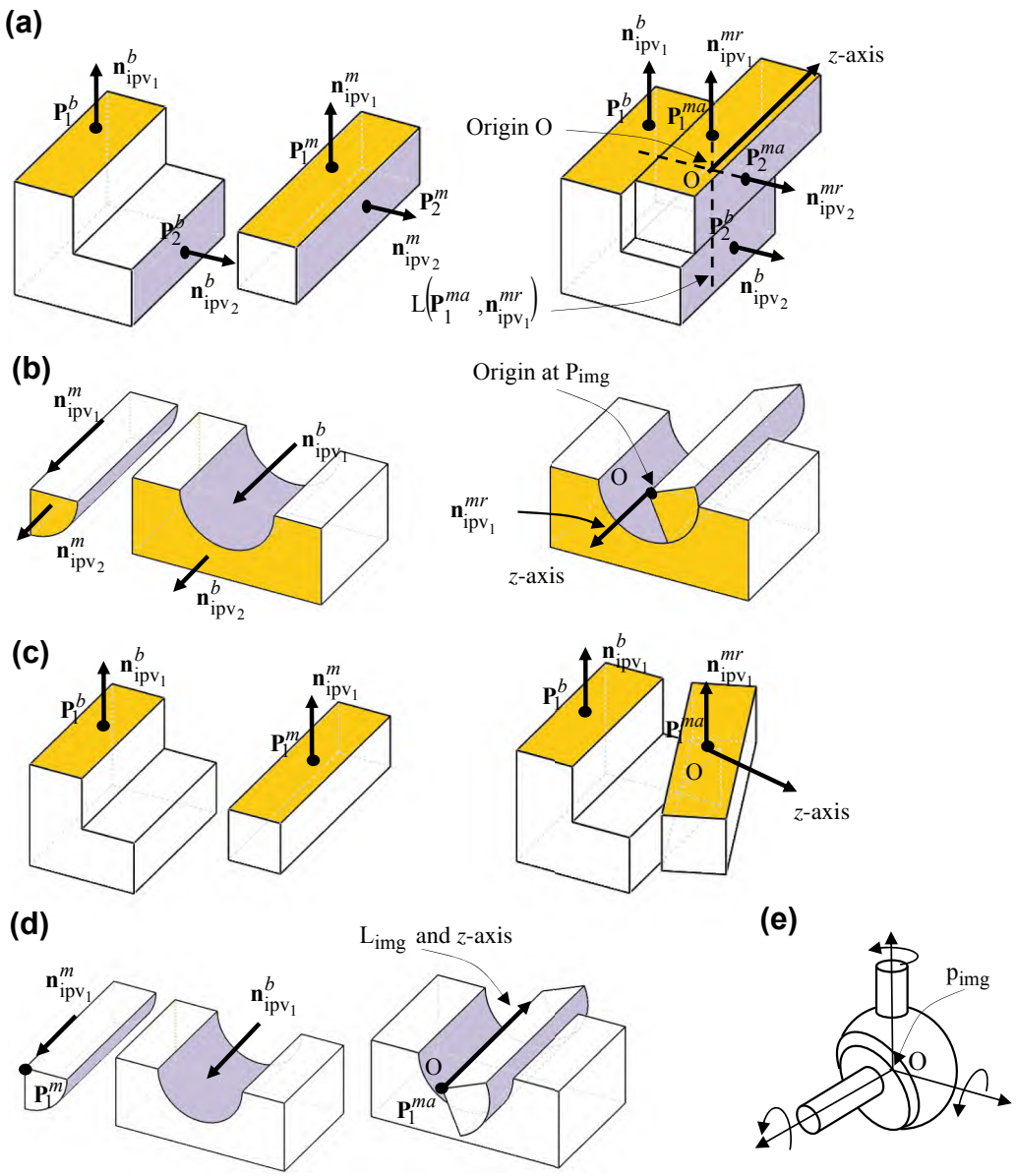


FIGURE 4.27

Determination of the origin and *z*-axis of a kinematic joint. (a) Prismatic joint, (b) revolute joint, (c) planar joint, (d) cylindrical joint, and (e) spherical joint.

The origin of the coordinate system can be determined, for example, by intersecting a line formed by $\mathbf{P}_1^{ma}$ and $\mathbf{n}_{\text{ipv}_1}^{mr}$ —that is, $\text{L}(\mathbf{P}_1^{ma}, \mathbf{n}_{\text{ipv}_1}^{mr})$—and a plane that is normal to $\mathbf{n}_{\text{ipv}_1}^{mr}$ and passes point $\mathbf{P}_2^{ma}$—that is, $\text{P}(\mathbf{P}_2^{ma}, \mathbf{n}_{\text{ipv}_1}^{mr})$, as shown in Figure 4.27(a). The z-axis of the prismatic joint is aligned with the moving axis and starts from the origin O. A similar arrangement can be made for other combinations of mating constraints that yield a prismatic joint.

For a revolute joint, the origin of the coordinate system O can be set at point of P_{img}, and the z-axis aligns with the moving axis, say $\mathbf{n}_{\text{ipv}_1}^{mr}$, as shown in Figure 4.27(b).

For a planar joint, the origin can be set at $\mathbf{P}_1^{ma}$, and the z-axis can be any vector on the plane, as shown in Figure 4.27(c).

For a cylindrical joint, the origin can be set at $\mathbf{P}_1^{ma}$, and the z-axis aligns with $\mathbf{n}_{\text{ipv}_1}^{mr}$, as shown in Figure 4.27(d).

For a spherical joint, the origin can be set at P_{img} (Figure 4.27(e)), and the z-axis can be arbitrarily chosen, for example, to align with the z-axis of the global coordinate system.

Note that the origin can also be placed at the origin of the local coordinate system of the base or mating part assigned by the CAD system.

In the following, we use the same slider-crank example to illustrate the steps of identifying z-axis and coordinate systems of joints from mating constraints. We first assume an open-loop system by removing the two coincident-aligned constraints: Coincident4 between the piston and rod, and Coincident5 between the piston and bearing shown in Figure 4.7(c). Note that if we add a parallel constraint between piston and bearing (Plane3@piston and Plane2@bearing), as shown in Figure 4.28(a), the mechanism is like that of the example shown in Figure 4.25 and is no-longer closed-loop. We assume the assembly is underconstrained by suppressing the mating constraint Coincident2 between Plane3 of the crank and Plane3 of the bearing, as shown in Figure 4.7(a).

In this example, the bearing is fixed to the inertial frame and is considered as a ground link, as shown in Figure 4.28(a). The initial configuration of the crank, rod, and piston is shown in Figure 4.28(b). We assume the same mating constraints for this example, as shown in Figure 4.7. We first illustrate the steps of determining IPV and IMG of each mating constraint, and the corresponding joints they represent. Then, we determine the z-axis and origin of the joint coordinate

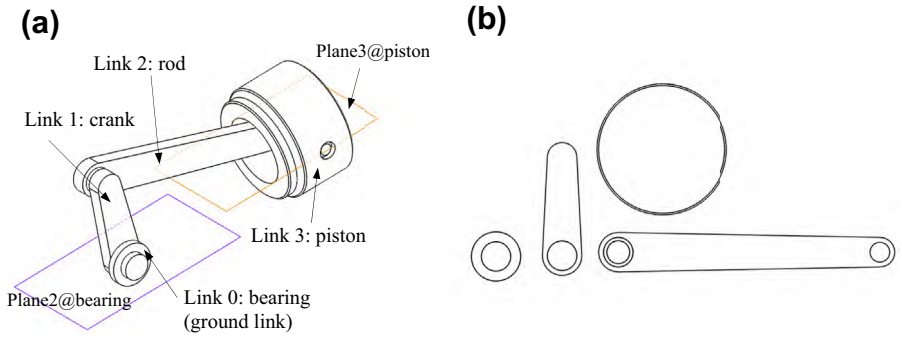


FIGURE 4.28

The open-loop slider-crank example. (a) The assembly in iso-view, and (b) the default configuration of the crank, rod, and piston (front view).

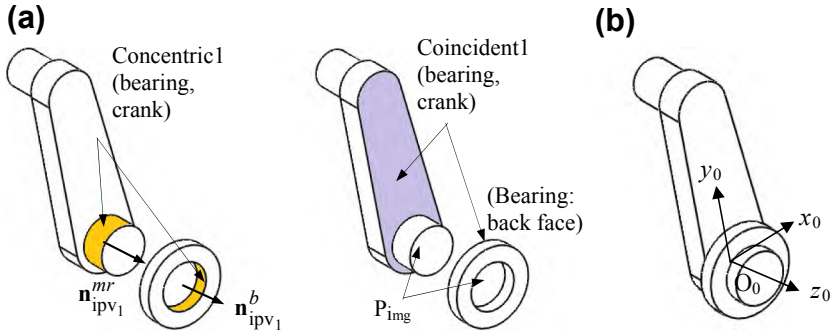


FIGURE 4.29

Joint origin and *z*-axis for the revolute joint between bearing and crank. (a) Mating constraints, IPV and IMG, and (b) coordinate system for the revolute joint.

systems for each joint. A kinematic model, represented in the D–H convention, can be constructed and the transformation matrices that position and orient the links can be computed using the approach discussed in Section 4.4.2.

The first two mates (*Concentric1* and *Coincident1*) assemble the crank to the fixed bearing, as shown in Figure 4.29(a). According to Figure 4.27(b), a revolute joint is extracted with the rotation axis z_0 pointing along a direction that aligns with $\mathbf{n}^{mr}_{\text{ipv}_1}$. Moreover, the origin of the coordinate system is located at the P_{img}. As a result, the origin O_0 is determined at the center of the hole of the bearing on the mating surfaces, as shown in Figure 4.29(b). The x_0 and y_0 axes are chosen conveniently, such as to align with the WCS, to form a right-hand frame.

The next two mates (*Concentric2* and *Coincident3*) assemble the rod to the crank, as shown in Figure 4.30(a). Again, a revolute joint is extracted, with the rotation axis z_1 pointing along a direction that aligns with $\mathbf{n}^{mr}_{\text{ipv}_1}$. Similarly, the origin O_1 is located at P_{img}, which is determined at the intersection of the axis of the upper shaft and the back face of the crank (i.e., on the mating surfaces between the rod and the crank), as shown in Figure 4.30(b). The coordinate system C_1 can be determined following the approach discussed earlier. The coordinate system C_1 aligns with C_0, except that it is offset along the y_0 direction by the amount that equals the length of the crank ℓ_1 and along the z_0-direction by an amount s_1, as shown in Figure 4.30(c). Similar to the crank, the rod is allowed to rotate with respect to the crank.

Now, we assemble the piston to the rod by adding *Concentric2* and *Coincident3* constraints, as shown in Figure 4.31(a). These two mating constraints create a revolute joint with the rotation axis z_2 pointing along a direction that aligns with $\mathbf{n}^{mr}_{\text{ipv}_1}$. Similarly, the origin O_2 is located at P_{img}, which is determined at the center of the circle at the midplane of the pin (on the mating surfaces between the rod and the piston), as shown in Figure 4.31(b). The coordinate system C_2 aligns with C_1, except that it is offset along the y_1 direction by the amount that equals the length of the rod ℓ_2 and along the z_2-direction by an amount s_2, as shown in Figure 4.31(c). The piston is allowed to rotate with respect to the rod.

Next, we add a coordinate system C_3 to the piston, as the end-effector, with its origin coinciding with that of C_2 and *z*-axis aligning with that of C_2. The coordinate system C_3 rotates a θ_3 angle along the z_3-axis, as shown in Figure 4.31(d). Note that in Figure 4.31(d), the position of the piston is lowered to simply better show the coordinate systems C_2 and C_3 as well as the rotation angle θ_3.

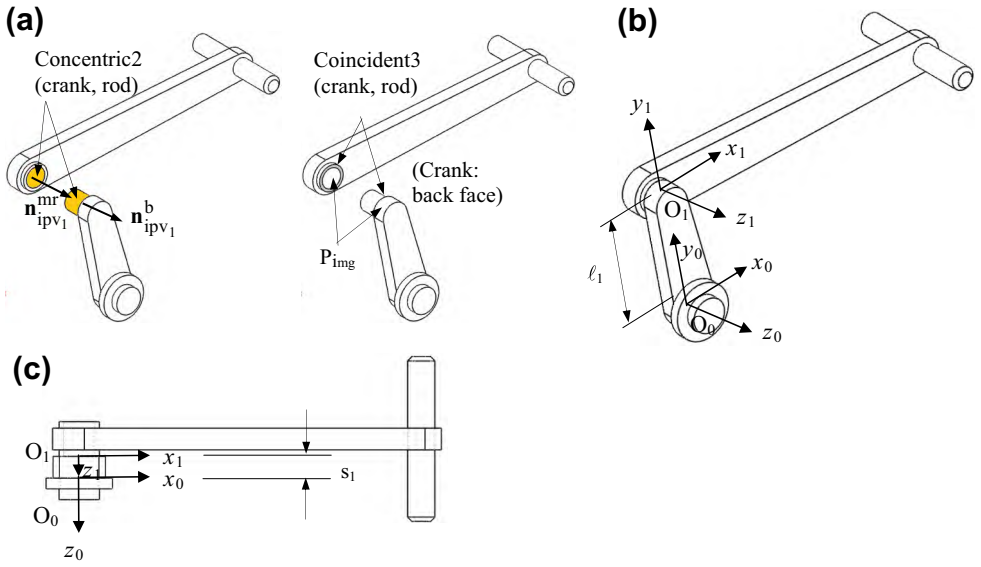


FIGURE 4.30

Joint origin and *z*-axis for the revolute joint between crank and rod. (a) Mating constraints, IPV and IMG, (b) coordinate system of the revolute joint, and (c) top view showing the offset s_1 between the origins.

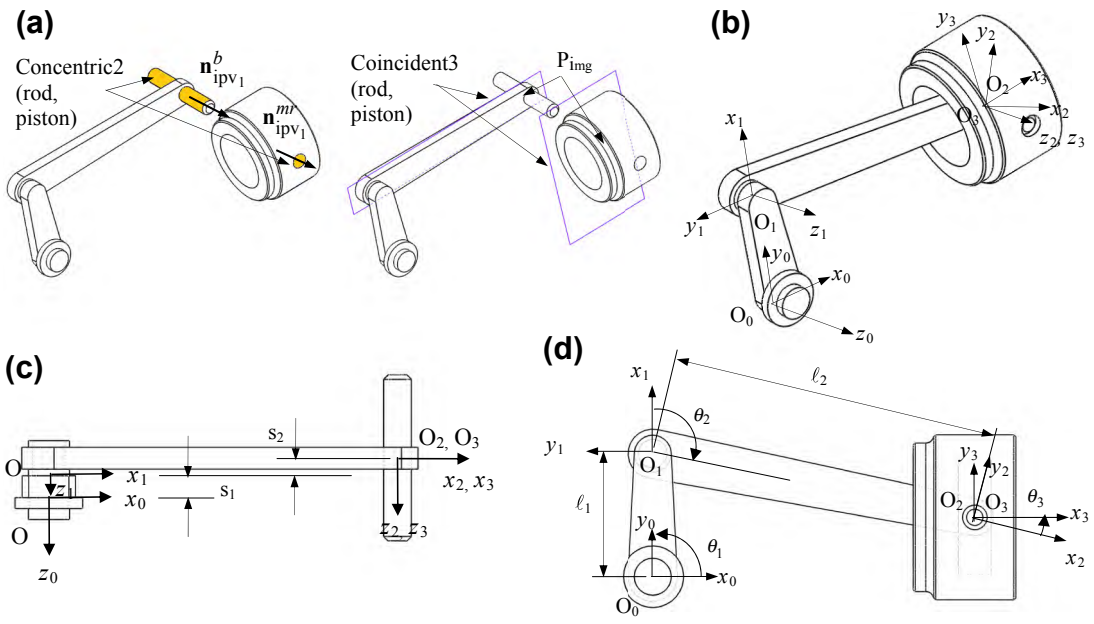


FIGURE 4.31

Coordinate systems C_2 and C_3. (a) Mating constraints, IPV and IMG, (b) coordinate systems in iso-view, (c) coordinate systems in top-view with offsets s_1 and s_2, and (d) coordinate systems in front-view.

Table 4.11 Link Parameters for the Open-Loop System of the Slider-Crank Mechanism

Link	θ_i	d_i	a_i	α_i
Crank (1)	θ_1	0	$a_1 = \ell_1$	0
Rod (2)	θ_2	$d_2 = -s_1$	$a_2 = \ell_2$	0
Slider (3)	θ_3	$d_3 = -s_2$	0	0

As a result, the table of link parameters for this open-loop system is created, as shown in Table 4.11. The transformation matrix for the mechanism can be constructed similar to that of Example 4.5, except that, in the current example, we have $d_2 = -s_1$ and $d_3 = -s_2$ (instead of $d_2 = -s$ and $d_3 = 0$).

Now, we discuss the closed-loop system. A prismatic joint is extracted by the mating constraints Coincident4 and Coincident5, in which the translational axis z_3 aligns with L_{img} formed by intersecting Front Plane@rod and Plane3@piston (or Plane2@piston and Plane3@bearing), as shown in Figure 4.32(a). For convenience, we pick the z_3-axis pointing to the right as positive, and place the

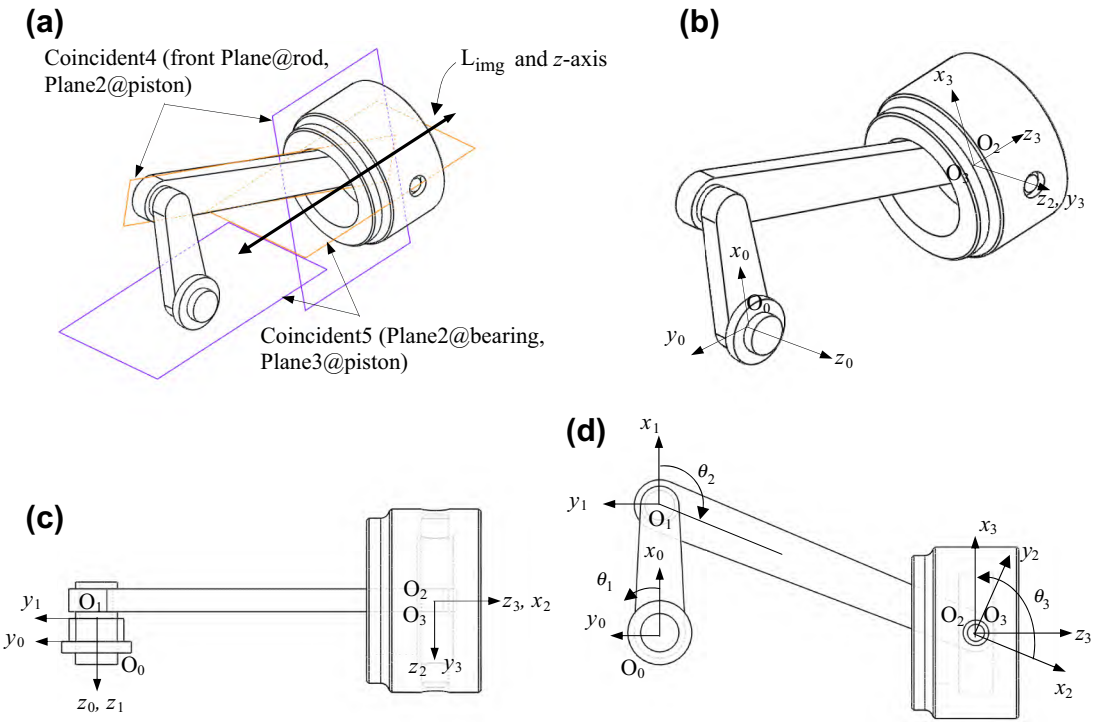


FIGURE 4.32

Determining the coordinate systems. (a) *Z*-axes, (b) coordinate systems, (c) origins of the coordinate systems (top view), and (d) origins of the coordinate systems (front view).

Table 4.12 Link Parameters for the Closed-Loop System of the Slider-Crank Mechanism

Link	θ_i	d_i	a_i	α_i
Crank (1)	θ_1	0	$a_1 = \ell_1$	0
Rod (2)	θ_2	$d_2 = -s_1$	$a_2 = \ell_2$	0
Slider (3)	θ_3	$d_3 = -s_2$	0	90°
Ground (0)	0	d_0	0	−90°

origin of the coordinate system at the same location as O_2, as shown in Figure 4.32(b). Because the axis z_3 is not in parallel with z_2, instead, they intersect; x_3 is determined pointing upward, and y_3 is also determined by the right-hand rule.

Now, the slider connects back to the ground. We must determine the x-axis of the coordinate system C_0. Just like that of the example shown in Figure 4.26, x_0 is determined to be pointing upward, and y_0 is also determined by the right-hand rule as shown in Figure 4.32(b). The coordinate systems determined are shown in the top and front views in Figure 4.32(c) and (d), respectively. Table 4.12 lists the link parameters for this closed-loop system. The transformation matrix for the mechanism can be constructed similar to that of Example 4.7, except that in the current system, we have $d_2 = -s_1$ and $d_3 = -s_2$, as shown in Table 4.12.

4.5 CASE STUDY AND TUTORIAL EXAMPLE

In this section, a case study and a tutorial example are presented. The case study presents briefly the applications of virtual reality technology to engineering design. The purpose of the case study is to showcase some of the interesting applications of CAD assembly to support engineering design. A single-piston engine is included as the tutorial example. Step-by-step instructions for creating the assembly model of the single-piston engine are given in Projects S1 and P1. Model files are available for download on this book's companion website (http://booksite.elsevier.com/9780123820389).

4.5.1 CASE STUDY: VIRTUAL REALITY

Virtual reality is the term used to describe a three-dimensional, computer-generated environment that can be explored and interacted with by a person. That person becomes part of this virtual world or is immersed within this environment and whilst there, is able to manipulate objects or perform a series of actions. One of the major development in virtual reality is CAVE (CAVE Automatic Virtual Environment), in which the person is fully immersed within it. CAVE takes the form of a cube-like space in which images are displayed by a series of projectors. Some systems enable the person to experience additional sensory input, such as sound or video, which contributes to the overall experience. A main feature of the CAVE system is interaction. The combination of interaction and total immersion is known as telepresence, in which a person can literally lose themselves within the virtual environment. Interaction takes place using a variety of input devices, such as a joystick, a wand or, more commonly, a haptics device (e.g., data glove). This enables the person to interact with objects, for example, by pulling, twisting, or gripping by means of touch. The ability to do this is known as haptics. An example

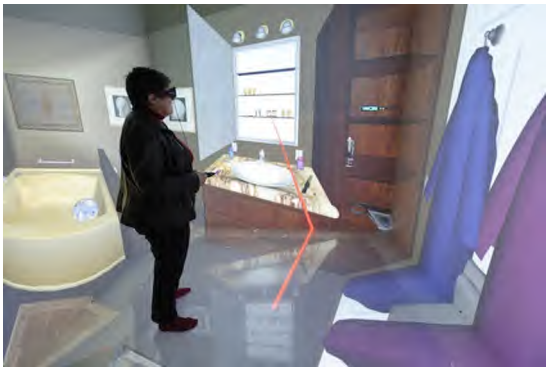


FIGURE 4.33

An example of CAVE system. (Figure courtesy of www.news.wisc.edu/21313.)

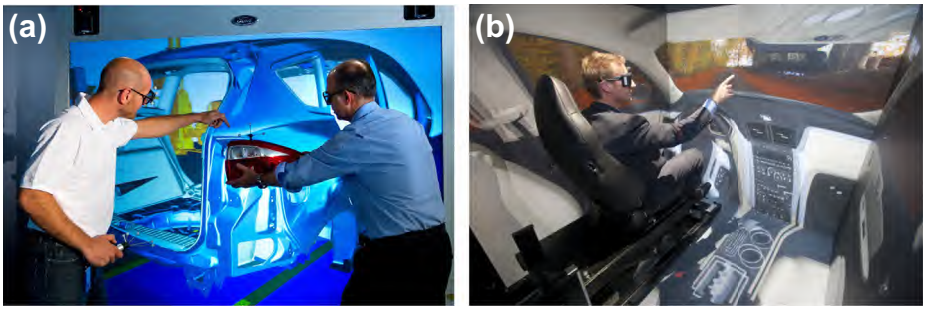


FIGURE 4.34

Applications of CAVE for engineering design. (a) Design evaluation of vehicle assembly at Ford (courtesy of www.enginetechnologyinternational.com/news.php?NewsID=41760), and (b) automotive interior design evaluation at GM (courtesy of www.parents.com/blogs/dadabase/2011/11/03/nostalgia/rise-of-the-dadmobile-the-chevy-traverse/).

of such a system is shown in Figure 4.33, in which a person wearing 3D eyeglasses and holding a virtual-reality controller steps into a room-sized, computer-generated version of a bathroom and interacts with medicine cabinet items and other moveable objects to simulate how patients self-administer health care at home (Taylor, 2012).

Virtual reality engineering includes the use of 3D modeling tools and visualization techniques as part of the design process. This technology enables engineers to view their project in 3D and gain a greater understanding of how it works. Plus they can spot flaws or potential risks before implementation. This also allows the design team to observe their project within a safe environment and make changes as necessary. What is important is the ability of virtual reality to depict fine-grained details of an engineering product to maintain the illusion. This means high-end graphics, video with a fast refresh rate, and realistic sound and movement. Automotive companies, such as Ford, uses CAVE (Figure 4.34(a)) to evaluate many aspects of the product design, including visibility, instrument

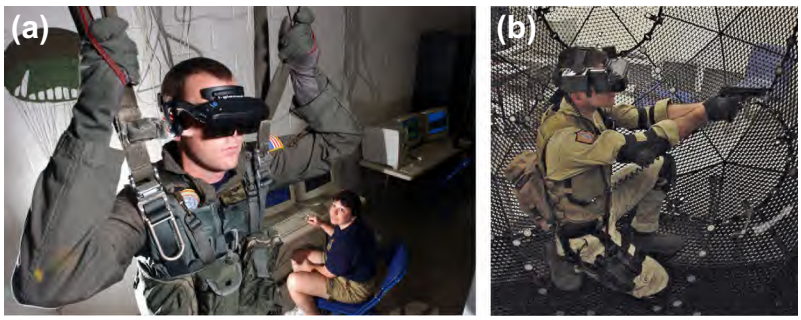


FIGURE 4.35

Examples of employing virtual reality for other applications. (a) Military training (courtesy of www.vrs.org.uk/virtual-reality-military/index.html), and (b) the first public virtusphere (courtesy of http://www.popsci.com/gadgets/article/2010-06/human-sized-hamster-ball-lets-you-play-virtual-worlds).

reach, ergonomics, and roominess before building a physical prototype (Engine Technology International, 2012). GM also uses CAVE to interact with the layout of the interior (Figure 4.34(b)), in which a designer wearing virtual reality glasses enters a small three-walled room where the proposed interior design of the vehicle is projected. The designer is immersed into the virtual interior of a vehicle that has not been built physically.

Virtual reality has been adopted by the military (this includes all three services—army, navy, and air force), where it is used for mainly training purposes. This is particularly useful for training soldiers for combat situations or other dangerous settings where they have to learn how to react in an appropriate manner. For example, a parachuting simulation can help train soldiers without flying them to 15,000 ft in the sky (Figure 4.35(a)). A virtual reality simulation enables them to do so but without the risk of death or serious injury. They can re-enact a particular scenario, such as engagement with an enemy in an environment in which they experience this, but without the real-world risks. Virtual reality has also been quickly adopted by the gaming industry. For example, the Excalibur Hotel and Casino in Las Vegas installed the first public Virtusphere, a human-sized hamster ball that lets players move through virtual worlds by walking, running, or crawling inside it (Duffy, 2010).

4.5.2 TUTORIAL EXAMPLE: A SINGLE-PISTON ENGINE

The engine example consists of four major components: case, propeller, connecting rod, and piston, as shown in Figure 4.36. In both SolidWorks and Pro/ENGINEER, the assembly of the example is organized as three subassemblies (*case_asm*, *propeller_asm*, and *connectingrod_asm*) and one part (*piston*). The *case_asm* is fixed. The *propeller_asm* is assembled to *case_asm* using concentric and coincident-mate constraints, as shown in Figure 4.37(a). The propeller is free to rotate along the x-direction. The *connectingrod_asm* is assembled to the propeller (at the crankshaft) using concentric and coincident-mate, as shown in Figure 4.37(b). The connecting rod is free to rotate relative to the propeller (at the crankshaft) along the x-direction. Finally, the piston is assembled to the connecting

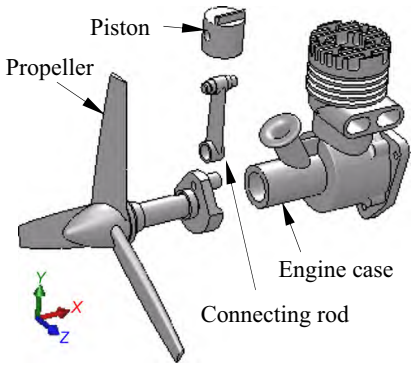


FIGURE 4.36

The single-piston engine example.

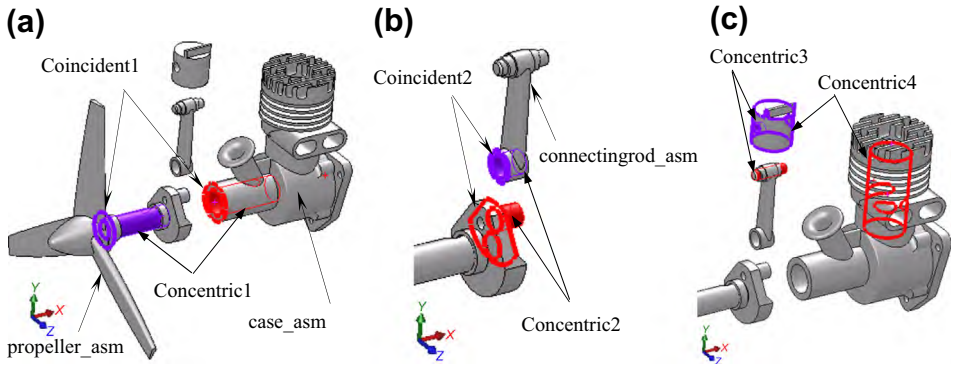


FIGURE 4.37

Assembly mating constraints defined for the engine example. (a) Mates between case and propeller, (b) mates between propeller and rod, and (c) mates between case and rod.

rod (at pin) using a concentric mate, as shown in Figure 4.37(c). The piston is also assembled to the case using another concentric mate. This mate restricts the piston movement along the *y*-direction, which in turn restricts the top end of the connection rod to move vertically.

4.6 SUMMARY

In this chapter, we discussed assembly modeling that supports CAD to represent an assembly. Topics such as mating constraints, kinematic joints, DOFs, and fully constrained vs underconstrained assemblies were discussed. We presented methods that support design changes and kinematic analysis in CAD assembly, which are the two most common activities encountered in assembly modeling

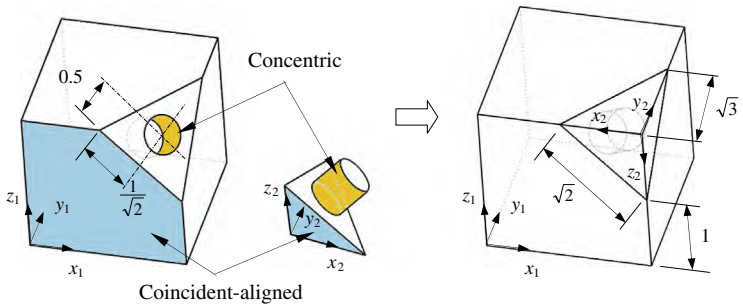
using CAD. In addition to theoretical discussion, we included virtual reality as a case study that illustrated the application of CAD assembly for practical engineering designs. In addition, a single-piston engine assembly was introduced as a tutorial example.

After going over this chapter, we hope you have a good understanding of the behind-the-scenes operations when you work on CAD assembly models. By this time, you should know how CAD determines the location and orientation of individual parts that constitute the assembly. In addition, how CAD handles design changes and supports kinematic analysis when you drag a component should be clear. You should know what is fully constrained and what is underconstrained while you are creating an assembly. When you encounter an error message, such as constraints in conflict, you should know precisely the internal algorithm that makes this call. We hope this chapter has been useful to you in obtaining a general understanding of the methods employed for assembly modeling in CAD, becoming familiar with the behind-the-scenes operations in CAD, and most importantly, making you more confident as a designer in creating and handling CAD assemblies in support of product development.

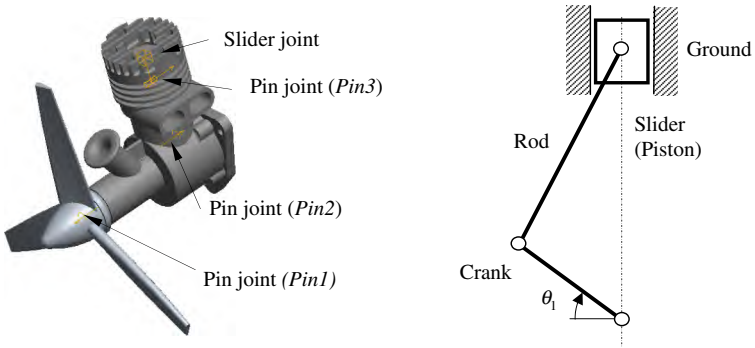
QUESTIONS AND EXERCISES

4.1. Calculate the transformation matrix for the example shown in Figure 4.17(b), assuming that the dimension d^b_{bottom} is changed to 6.

4.2. Calculate the transformation matrix for the example shown below. The base part (left) is a $2 \times 2 \times 2$ cube with the top right corner removed and a blind hole of diameter 0.5 drilled. The position of the hole is also shown. The mating part (shown below, center) is a tetrahedral block with a short cylinder, which simply complements the cut-out portion of the base part. Two mating constraints, concentric and coincident-aligned, are employed to create an assembly shown below (right).



4.3. Define coordinate systems and calculate the transformation matrix for the single-piston engine mechanism shown on the next page (left). There are four joints, a pin joint (*Pin1*) between the propeller and case, the second pin joint (*Pin2*) between the connecting rod and the crankshaft (propeller), a third pin between the piston and the piston pin (mounted on the connecting rod), and a slider joint between the piston and the case. Kinematically, the system is a planar four-bar linkage shown on the next page (right), consisting of four links: crank, rod, slider, and ground. Note that the lengths of the rod and crank are 2.25 and 0.58333, respectively.



4.4. Continue with Problem 3 and answer the following questions by formulating and solving the equations involved.
- **a.** If $\theta_1 = 0°$, calculate all link parameters that determine the configurations of the system.
- **b.** Change the lengths of the crank and rod to 1 and 3, respectively; and calculate the parameters that determine the location and orientation of individual parts. First (Part A), we assume that the angle θ_1 is $\theta_1 = 0$. Then, in Part B, we assume the piston is stationary.
- **c.** Change the angle θ_1 to 45° (Part A) and then 180° (Part B), and calculate the parameters that determine the configurations of the system.

4.5. Conduct a case study in the application of virtual reality (or CAVE) technology for engineering design that was not included in Section 4.5. In your one-page report, please include the following:
- **a.** Name of the company or organization
- **b.** Source of the information (article, paper, magazine, website, YouTube, etc.)
- **c.** What is the nature of the application? What kind of equipment is employed to support such an application? What is the value added to the company or organization by employing the virtual reality technology?

REFERENCES

Adams, J.D., Gerbino, S., Whitney, D.E., July 1999. Application of screw theory to motion analysis of assemblies of rigid parts. In: Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning Porto, Portugal, pp. 75–80.

Chang, K.H., 2010. Motion Simulation and Mechanism Design Using SolidWorks Motion 2011. SDC Publications.

Craig, J.J., 1989. Introduction to Robotics: Mechanics and Control, second ed. Addison-Wesley.

Dawari, A., Sen, D., December 12–13, 2007. Relation of Part Positions and Kinematic Freedom: a Survey, 13th National Conference on Mechanisms and Machines (NaCoMM07). IISc, Bangalore, India.

Denavit, J., Hartenberg, R.S., 1955. A kinematic notation for lower-pair mechanisms based on matrices. Transaction ASME Journal of Applied Mechanics 23, 215–221.

Duffy, J., June 23, 2010. Human-Sized Hamster Ball Lets You Play in Virtual Worlds. Popsci. http://www.popsci.com/gadgets/article/2010-06/human-sized-hamster-ball-lets-you-play-virtual-worlds.

Eng, T.-H., Ling, Z.-K., Olson, W., McLeanb, C., 1999. Feature based assembly modeling and sequence generation. Computerised Industrial Engineering 36, 17–33.

Erdman, A.G., Sandor, G.N., Kota, S., 2001. Mechanism Design: Analysis and Synthesis, fourth ed. Prentice Hall.
Ford Starts Work on Virtual Assembly Lines, August 09, 2012. Engine Technology International. http://www.enginetechnologyinternational.com/news.php?NewsID=41760.
Hartenberg, R.S., Denavit, J., 1965. Kinematic Synthesis of Linkages. MCGraw-Hill Series in Mechanical Engineering. McGraw-Hill, New York, p. 435.
Joskowicz, L., 1990. Mechanism comparison and classification for design. Artificial Intelligence in Engineering and Design 2 (1), 149–166.
Kim, S.H., Lee, K., 1989. An assembly modeling system for dynamic and kinematic analysis. Computer Aided Design 21 (1).
Kim, M.G., Wu, C.-H., November 1990. A formal part mating model for generating compliance control strategies of assembly operations. In: Los Angeles, CA, USA IEEE International Conference on Systems, Man and Cybernetics, pp. 611–616.
Kim, J., Kim, K., Choi, K., Lee, J.Y., 2000. Solving 3D geometric constraints for assembly modelling. International Journal of Advanced Manufacturing Technology 16, 843–849.
Kim, J., Kim, K., Lee, J.Y., 2001. Solving 3D Geometric Constraints for Closed-Loop Assemblies. In: The 5th International Conference on Engineering Design & Automation, pp. 368–373. Las Vegas.
Kim, K.-J., Sacks, E., Joskowicz, L., 2003. Kinematic analysis of spatial fixed-axis higher pairs using configuration spaces. Computer Aided Design 35, 279–291.
Kim, J.S., Kim, K.S., Lee, J.Y., Jung, H.B., 2004. Solving 3D geometric constraints for closed loop assemblies. International Journal of Advanced Manufacturing Technology 23, 755–761.
Kim, J.S., Kim, K.S., Lee, J.Y., Jeong, J.H., 2005. Generation of assembly models from kinematic constraints. International Journal of Advanced Manufacturing Technology 26, 131–137.
Lee, K., Andrews, G., 1985. Inference of the positions of components in an assembly part: 2. Computer Aided Design 17 (1), 20–24.
Lombard, M., March 2013. SolidWorks 2013 Bible. Wiley.
Lozano-Pérez, T., 1983. Spatial planning: a configuration space approach. IEEE Transactions Computing 32 (2), 108–120.
Reyes, A., 2013. Beginner's Guide to SolidWorks 2013-Level I. SDC Publication.
Shih, R.H., 2013. Parametric Modeling with Pro/ENGINEER Wildfire 5.0. SDC Publication.
Singh, P., Bettig, B., 2004. Port-compatibility and connectability based assembly design. Journal of Computers, Informatics, Science and Engineering 4.
Sinha, R., Gupta, S.K., Paredis, C.J.J., Khosla, P.K., 2002. Extracting articulation models from CAD models of parts with curved surfaces. Journal of Mechanical Design 124 (1), 106–114.
Spoong, M.W., Hutchinson, S., Vidyasagar, M., 2005. Robot Modeling and Control. Wiley.
Taylor, C., December 4, 2012. Virtually Healthy: 'CAVE' Lets Researchers Experience Patients' Behavior. University of Wisconsin-Madison News. www.news.wisc.edu/21313.
Toogood, R., Zecher, J., 2012. Creo Parametric 1.0 Tutorial and MultiMedia DVD. SDC Publication.
Whitney, D.E., 2004. Mechanical Assemblies Their Design, Manufacture, and Role in Product Development. Oxford University Press, New York.

CHAPTER

DESIGN PARAMETERIZATION

5

CHAPTER OUTLINE

5.1 Introduction 234
5.2 Design Intents 235
5.3 Design Axioms 238
5.3.1 Independence Axiom 238
5.3.2 Information Axiom 241
5.4 Design Parameterization at Part Level 242
5.4.1 Profile in Sketch 242
5.4.2 Solid Features in Part 244
5.4.3 Guidelines for Design Parameterization 245
5.5 Design Parameterization at Assembly Level 248
5.5.1 Guidelines for Design Parameterization 248
5.5.2 Slider-Crank Assembly in Pro/ENGINEER 250
5.5.3 Slider-Crank Assembly in SolidWorks 252
5.6 Case Studies 253
5.6.1 Single-Piston Engine 253
5.6.1.1 Part Level: Engine Case 255
5.6.1.2 Assembly Level: Engine 256
5.6.2 HMMWV Suspension 257
5.6.2.1 Track Design Variable 258
5.6.2.2 Wheelbase Design Variable 260
5.6.2.3 Design Change 261
5.7 Summary 262
Questions and Exercises 262
References 263

Design changes are frequently encountered in the product development process. The complexity of the design change is multiplied when the product design involves large-scale assemblies with multiple engineering disciplines. Very often, a simple change in one part may propagate to its neighboring parts, therefore affecting the entire product assembly. Both parts and assembly must be regenerated (or rebuilt) for a valid product model. At the same time, the regenerated product model must satisfy the geometric design requirements and meet the designer's expectations or intents.

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00005-3
Copyright © 2015 Elsevier Inc. All rights reserved.

When a product is being developed in a virtual environment, the design changes are often implemented first by altering the geometry of the product represented in solid models using a computer-aided design (CAD) tool. If the product solid model is not parameterized properly, the changes in geometry often lead to invalid parts or assembly. At part level, the changes may yield a solid model with invalid solid features if it is not properly parameterized. In this case, the entire product assembly is in vain. Even when individual parts of the product are regenerated correctly, parts may still penetrate to their neighboring parts or leave excessive gaps between them, if the solid model is not properly parameterized at the assembly level.

In this chapter, the fundamental principles of design parameterization for parts and assembly will be discussed. A set of guidelines will be presented for designers to parameterize solid models in order to capture design intents more effectively. These guidelines, which are provided at both part and assembly levels, support designers in successfully conducting product design in the e-Design environment.

A number of simple examples are included to explain concepts and methods. A slider-crank mechanism and its crankshaft are employed to illustrate and demonstrate the practicality of guidelines developed for both Pro/ENGINEER and SolidWorks. In addition, a single-piston airplane engine and a high-mobility multipurpose wheeled-vehicle (HMMWV) suspension are presented as case studies to demonstrate the parameterization method for practical applications. Note that in this chapter, parts and assembly are created in respective CAD tools. Issues of solid model translations between CAD systems will be addressed in Chapter 6. CAD models of the examples employed in this chapter can be found on the book's companion site. More detailed instructions for bringing up these models and steps for carrying out studies described in this chapter can be found in Projects P1 and S1 of the book.

Overall, the objectives of the chapter are to (1) introduce fundamental principles of design parameterization for designers to capture design intents at both part and assembly levels, and (2) offer practicing guidelines and illustrations for designers to facilitate the construction of parametric solid models. Note that the guidelines provided in this chapter may be used to support design practice and can be extended as needed.

5.1 INTRODUCTION

After intensive research and development in recent decades, the feature-based parametric modeling technique has become a reality (Lee, 1999; Zeid, 1991). This technique has been widely adopted in the mainstream CAD tools, such as Pro/ENGINEER, SolidWorks, SolidEdge, Unigraphics, CATIA, and even Mechanical Desktop of AutoCAD. With such a technique, designers are able to create parts through solid features and assemble parts or subassemblies for a complete product digital mockup in the CAD environment. In addition, the designer is able to define design variables by relating dimensions of part features and create assembly mating constraints between parts to parameterize the product model through the parametric modeling technique. With the parameterized product model, the designer can make a design change simply by changing geometric dimension values and asking the CAD software to automatically regenerate the parts that are affected by the change, and hence the entire assembly.

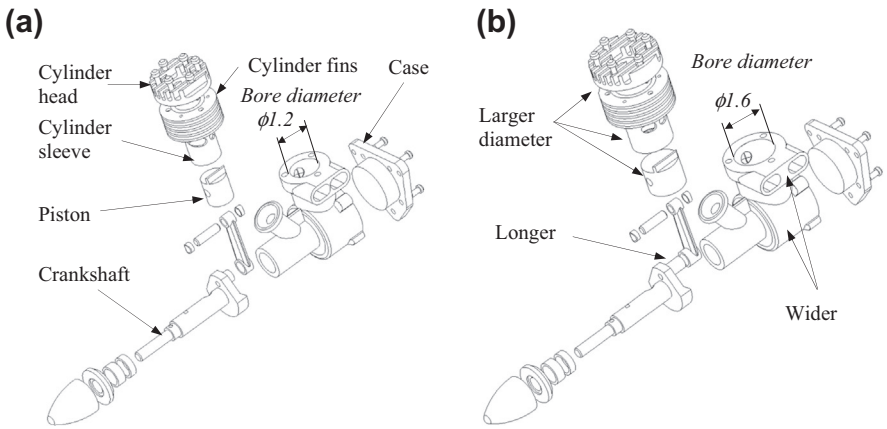


FIGURE 5.1

An exploded view of a single-piston engine with a bore diameter of 1.2 in. (a) and a bore diameter of 1.6 in. (b).

For example, the bore diameter of an engine case is defined as a design variable, as shown in Figure 5.1(a). When the diameter is changed from 1.2 in. to 1.6 in., the engine case is regenerated first by properly updating its solid features that are affected by the change. As shown in Figure 5.1(b), the engine case becomes wider and the distance between the two exhaust manifolds is larger, just to name a few. At the same time, the change propagates to other parts in the assembly, including the piston, piston pin, cylinder head, cylinder sleeve, cylinder fins, and crankshaft, as illustrated in Figure 5.1(b). More important, the parts stay intact, maintaining adequate assembly mating constraints, and the change does not induce interference nor leave excessive gaps between parts. With such parametric models, designers are given tremendous freedom to explore design alternatives efficiently and accurately. In addition, this parametric technology supports the cross-functional team in conducting parametric studies and designing trade-offs in the e-Design environment (Chang et al., 1999). More about parametric study and design trade-off methods are discussed in Chapter 17 Design optimization.

We start in Section 4.2 by introducing design intents in product solid models. With the understanding of design intents, we discuss the two design axioms in Section 4.3 that form the basis of the design parameterization methods to be discussed in this chapter. In Sections 4.4 and 4.5, we offer guidelines for design parameterization at part and assembly levels, respectively. Section 4.6 includes two case studies, an airplane engine and an HMMWV suspension, which demonstrate the application of the parameterization method and guidelines to practical examples.

5.2 DESIGN INTENTS

In a broader scope, design intent (DI) is a realization of design requirements (DR) in the shape of the product solid model. In the context of the e-Design paradigm, design intent is defined as the geometric shape of parts and/or configuration of the product assembly that the designer desires to attain while changing dimension values of the product solid model in CAD for exploring better design alternatives.

In practice, design intents are derived from design requirements, satisfying physical, functional, or performance requirements through the geometric shape of the product solid model. These design intents must be implemented in CAD for the purpose of exploring design alternatives that reveal better product performance and meet the design requirements.

While exploring design alternatives, changes are realized in CAD by modifying geometric dimension values and regenerating (or rebuilding) the product solid models automatically. In order to capture the DI, the product solid model must be properly constructed and parameterized.

Design intent for a single part can be captured by properly creating individual solid features and carefully relating dimensions within or between features so that when a dimension value is changed, the solid features affected by the change can be regenerated or rebuilt successfully. The geometric dimensions that can be changed independently to capture design intents are called design variables (DVs). The relationships between DR, DI, and DV are shown in Figure 5.2(a). Figure 5.2(b) depicts an uncoupled design, in which one design requirement is realized in two DIs. DI1 is implemented in CAD in such a way that one design variable DV1 (an independent dimension) affects the DI. On the other hand, DI2 is captured by two design variables, DV2 and DV3. A change in DV1 does not affect DI2, and changes in DV2 or DV3 do not affect DI1. Figure 5.2(c) illustrates a coupled design intent, in which a change in DV2 affects both DIs.

In general, an uncoupled design is much more desirable than a coupled one. In some cases, a coupled design intent may be decoupled by adding more DIs and/or DVs, which will be illustrated later in Section 5.3. The design requirements depicted in Figures 5.2(b) and (c) are decoupled. It is possible that DRs may be coupled, in which one DI may affect both DRs. Note that in this chapter, we focus more on capturing DIs and less on DRs, except for the slider-crank example to be discussed in Section 5.3.1.

To illustrate more, we use a block with a hole, which is shown in Figure 5.3, as an example. This part consists of a base extrusion block and a through hole. The design intent derived from the design requirement is to keep the hole right at the center of the block while varying its size. It is apparent that the DI is uncoupled because it is the only DI being considered.

To capture the intent for this simple example, relations between dimensions must be created between the hole's centerpoint and the size of the block. As shown in Figure 5.3, the relations are $d1 = d2/2$, and $d3 = d4/2$, where $d2$ (block width) and $d4$ (block height) are design variables.

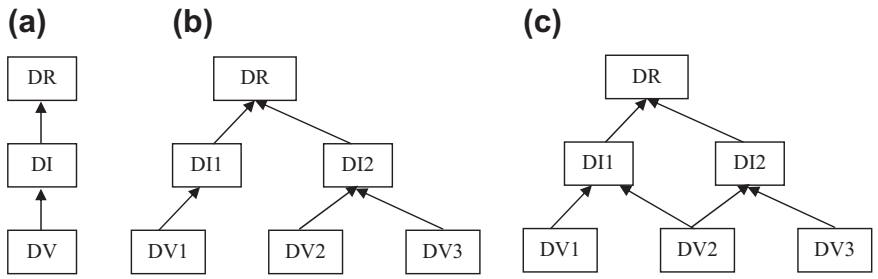


FIGURE 5.2

Illustration of the relationships between design requirement (DR), design intent (DI), and design variables (DV). (a) Relationship between DR, DI, and DV. (b) Uncoupled design intent. (c) Coupled design intent.

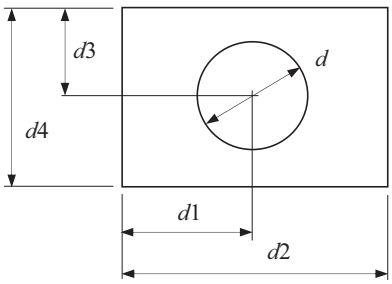


FIGURE 5.3

The example of a block with a center hole.

Before defining the relations, a rectangular profile was first created in the sketch before extruding the base block feature. In the rectangular profile, several sketch relations were imposed so that the size of the block can be determined by the width and height dimensions only (i.e., $d2$ and $d4$). The sketch relations specify that the top and bottom edges of the rectangle are horizontal, the left and right edges are vertical, and one of the corner points is anchored to the coordinate system of the sketch plane. As a result, the sketch relations and dimensions fully constrain the profile in the sketch. Once the base block is extruded, the center hole can be created as, for example, an extruded cut feature. Details about the sketch and sketch relations were discussed in Chapter 3.

Note that it is important to also capture the upper and lower limits of a design variable. In general, these limits must ensure a design that is always physically meaningful. In addition, the limits must ensure that a valid solid model can be regenerated in CAD. For the example shown in Figure 5.3, the lower limits of the width ($d2$) and height ($d4$) must be greater than the hole diameter 'd' for a physically meaningful design. However, if the concern is simply about the solid model regeneration, both variables should at least be greater than zero.

At the assembly level, DI is captured by defining adequate mating constraints and relating dimensions across parts so that a change in dimension value can be propagated to all parts affected. The parts affected must be regenerated successfully; at the same time, they must maintain proper positions and orientations with respect to one another without violating any mating constraints nor revealing part penetration or excessive gaps between them. Moreover, the regenerated solid model must meet the designer's expectations. The bracket assembly shown in Figure 5.4 illustrates these points.

The bracket assembly consists of four parts: bracket, bushing, shaft, and arm. Mating constraints, such as concentric and surface mate, are defined to assemble the parts. In addition, one relation is defined to relate the shaft diameter and inner diameter of the hole in the bushing to ensure that the assembly is properly retained when the shaft size is changed. The relation is defined as

$$\phi d1{:}2 = \phi d3{:}4 + 0.01 \tag{5.1}$$

where $\phi d1{:}2$ is the diameter of the hole in the bushing, $\phi d3{:}4$ is the outer diameter of the shaft, and 0.01 is the prescribed clearance between them. Note that in Eq. 5.1, $\phi d1{:}2$ becomes a dependent dimension and $\phi d3{:}4$ stays independent.

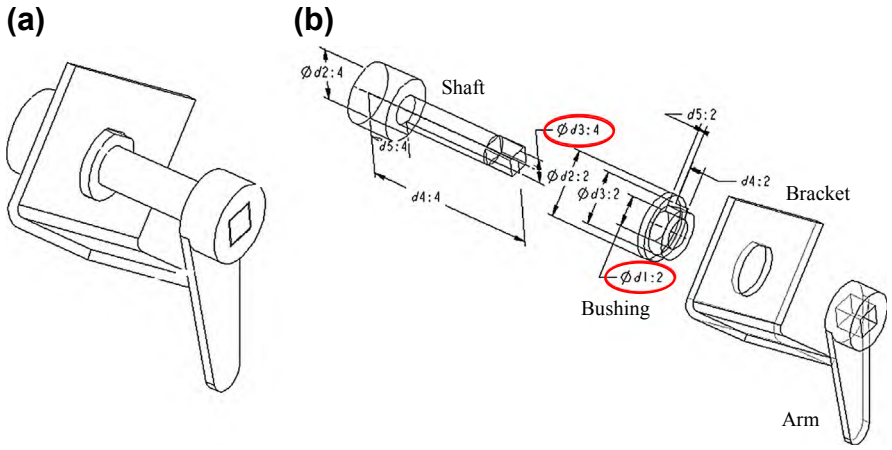


FIGURE 5.4

The bracket assembly example. (a) Unexploded view. (b) Exploded view.

5.3 DESIGN AXIOMS

As mentioned earlier, a set of guidelines for design parameterization will be presented. These guidelines were developed following two important axioms from Suh (1990):

Axiom 1: The Independence Axiom (maintains the independence of design intent)
Axiom 2: The Information Axiom (minimizes the information content of the design intent)

Axiom 1 implies that changing the DV values has an effect only on the referent DI. In other words, it is desirable to create uncoupled DIs whenever possible. Because DIs are derived from design objectives, Axiom 1 is often exercised to address coupled design objectives in practice.

Axiom 2 states that the amount of information (usually number of DVs) that is available to the designer for making design changes must be minimized for each DI.

5.3.1 INDEPENDENCE AXIOM

Generally speaking, the independence axiom is easier to comply with than that of the information axiom. Often the challenge lies in deriving uncoupled design intents from (sometimes) coupled design objectives or potentially conflict design requirements. It may not always be possible to create uncoupled DIs. When coupled DIs are unavoidable, additional DIs (sometimes DVs) may have to be added in order to resolve or alleviate the conflict. In this case, the DIs are referred to as *decoupled*. A decoupled DI is less desirable, it is simply an unavoidable compromise.

An uncoupled design is always superior to a coupled or decoupled one. This is because the DIs in the uncoupled design can be attended much easier, especially for complex design problems with large-scale assembly, because the effect of individual DV on the referent DI is completely separated. Moreover, an uncoupled design often carries less information for the designer to attend.

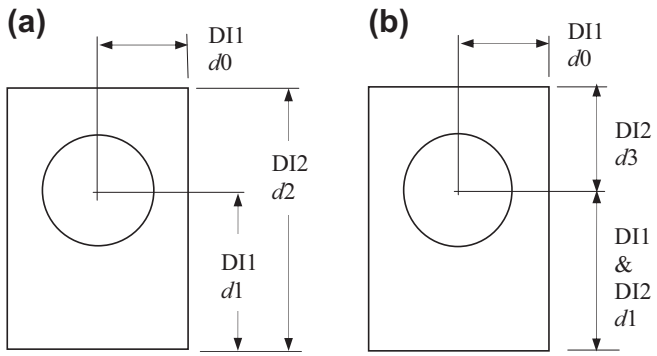


FIGURE 5.5

Illustration of design intents (DIs). (a) Uncoupled DIs. (b) Coupled DIs.

A simple example of an uncoupled design is shown in Figure 5.5(a). The design is of a plate with an orifice used to measure flow rates. The design requirement is simply positioning the hole inside the plate for a physically viable design. We assume two DIs in this case:

DI1: position of the orifice (i.e., $d0$ and $d1$), which keep the orifice completely inside the plate.
DI2: height of the plate (i.e., $d2$), which is sufficiently large to enclose the orifice.

The design in Figure 5.5(a) is uncoupled because perturbation in the value of $d0$ and $d1$ that defines the position of the hole has no effect on the height of the plate.

The same plate example may be created as a coupled design, where the height of the block is determined by the sum of the two DVs, d1 and d3, as shown in Figure 5.5(b). Perturbing d1 not only alters the orifice position, but it also affects the height of the plate. For the two DIs to remain independent, they need to be referenced to a datum that is not a DV, such as the dimensions shown in Figure 5.5(a), where the bottom edge of the plate serves as the datum. Both solid models in Figures 5.5(a) and (b) are valid. The solid model in Figure 5.5(a) provides the designer with a clearer perspective on how each DV affects its own DI. Although the solid model in Figure 5.5(b) is valid, it does not comply with the independence axiom; therefore, its use in the design process may be cumbersome.

A slider-crank example shown in Figure 5.6 is presented to further illustrate the issues involved in the design parameterization. This slider-crank mechanism consists of four parts: crankshaft, connecting rod, piston pin, and piston. Two design requirements are defined in this example:

DR1: Horizontal velocity of the piston increases 20% when the crankshaft is driven at the same angular velocity.
DR2: Weight of the mechanism reduces 5%.

It is apparent that both DRs are affected by the top center position of the mechanism, assuming the mechanism as part of a single piston engine. Therefore,

DI1: top center position, which is realized by the lengths of the crankshaft and connecting rod. Moreover, the first design requirement can be achieved, for example, by increasing the length of the crankshaft $d2{:}0$ (or the length of the connecting rod $d3{:}2$), as shown in Figure 5.6(b). The dimension $d2{:}0$ becomes the DV of the first design intent. However, changing the DV also affects

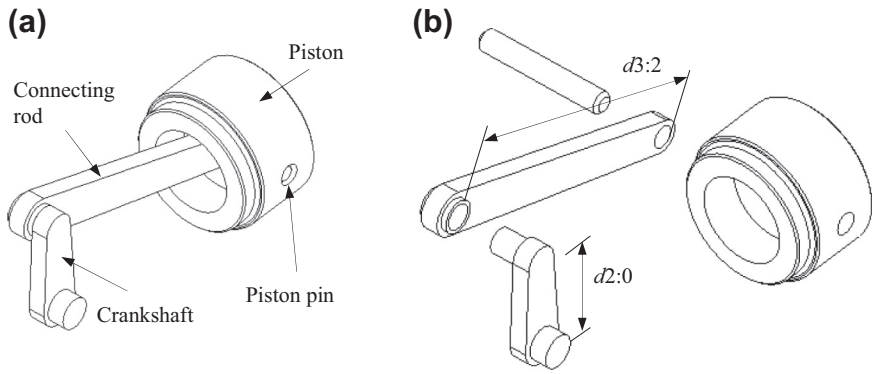


FIGURE 5.6

The slider-crank mechanism. (a) Unexploded view. (b) Exploded view.

the second design requirement—the weight of the mechanism. In this case, these two design requirements are coupled.

In order to reduce the coupling effect, a second DI can be defined that, for instance, reduces the width of the connecting rod:

DI2: Width of the connecting rod.

This will help achieve the second design requirement or alleviate the effect of the change in $d2{:}0$ on the second design requirement, the weight. Adding the second DI helps to decouple the design requirement and, therefore, better comply with the independence axiom.

This is what Axiom 2 (information axiom) asks. The crankshaft and connecting rod must be properly parameterized (more to be discussed in Section 5.4.3) in order to capture the design variables. When a DV is changed, the change must be propagated to the affected parts. The remaining parts must be kept unchanged, and the entire assembly must be maintained intact, as illustrated in Figure 5.7.

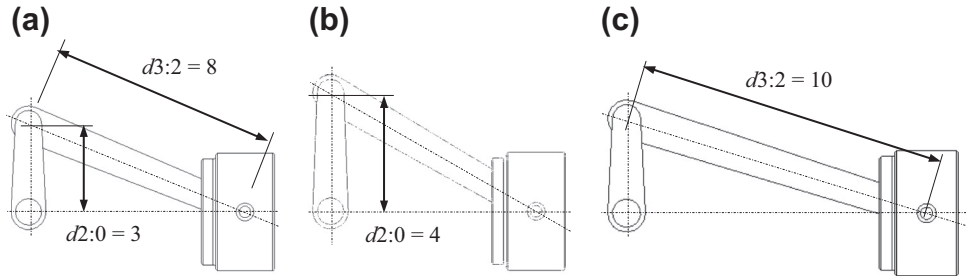


FIGURE 5.7

Changes of the length of the design variables. (a) Design variables $d2{:}0$ and $d3{:}2$. (b) $d2{:}0$ changes to 4. (c) $d3{:}2$ changes to 10.

5.3.2 INFORMATION AXIOM

The second axiom, the information axiom, can be primarily addressed in the sketch of the solid feature. A bracket example created in Pro/ENGINEER is shown in Figure 5.8. The bracket profile consists of two horizontal and three vertical line segments, two perpendicular line segments, two quarter circular arcs, and two circles. By using the Intent Manager of Pro/ENGINEER (Toogood and Zecher, 2012), the profile is fully constrained with ten dimensions and a number of sketch relations, including vertical (V), horizontal (H), tangent (T), concentric (⊕), and vertical alignment (❙) (see Figure 5.8(a)). The symbols of the geometric constraints that appear in Figure 5.8(a) were explained in Chapter 3.

Note that the information contents of the sketch profile shown in Figure 5.8(a) are not minimized. Assume that a DI is to keep the profile symmetric with respect to its middle horizontal line. In order to capture the symmetry DI, entities must be related. For example, the radii of the two circular arcs, the radii of the circles, and lengths of the two vertical line segments must be changed simultaneously. Keeping the sketch profile as it is and creating dimension relations to capture the DI is complex and unnecessary. A better option is to add sketch relations to properly parameterize the profile. While adding sketch relations, redundant or conflict dimensions will be removed by CAD automatically.

As shown in Figure 5.8(b), two equal radii constraints (R_1–R_1 and R_2–R_2), three equal lengths constraints (L_1–L_1, L_2–L_2, and L_3–L_3), and a perpendicular constraint (⊥) are added. As a result, the symmetry DI is properly captured and the number of dimensions is reduced to four, as shown in Figure 5.8(b). Changes in any of the dimensions yield a symmetric sketch with respective to its middle horizontal line. Note that the profile in Figure 5.8(b) is simpler and complies better with the information axiom.

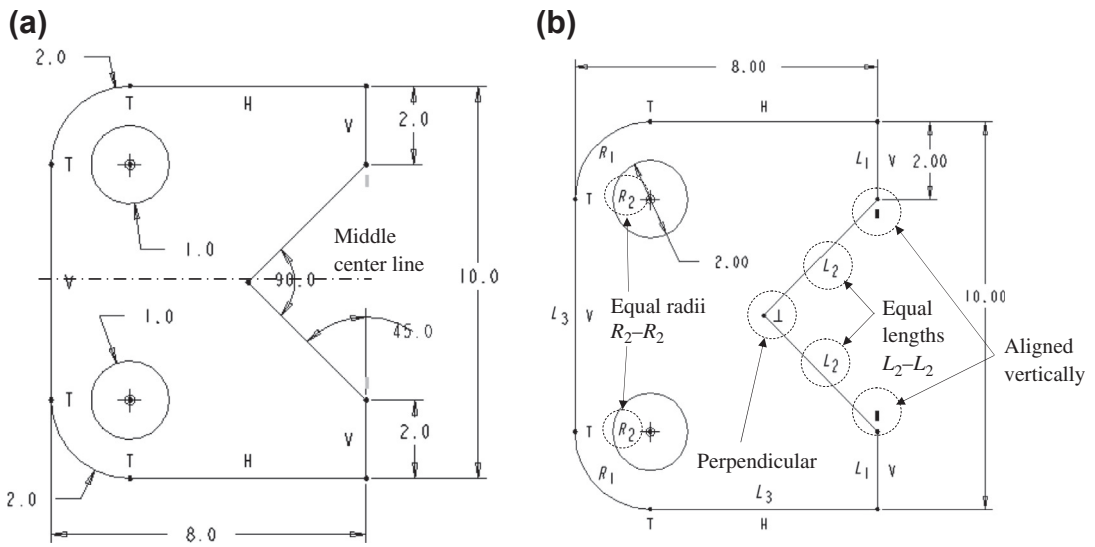


FIGURE 5.8

Minimization of information contents. (a) Information not minimized. (b) Information minimized.

The profile shown in Figure 5.8(b) can also be created in SolidWorks because similar sketching capabilities and sketch relations are available in SolidWorks sketch mode. Commonly employed sketch relations in both Pro/ENGINEER and SolidWorks were summarized in Appendix 3A in Chapter 3.

In addition to the sketch profile, relations in Pro/ENGINEER or equations in SolidWorks can be added to relate dimensions between features. When new features are created by copying, mirroring, or patterning existing features, additional dimensions may be assigned as dependent on those of the original feature in order to reduce the information content, if it is consistent with the DI to capture.

5.4 DESIGN PARAMETERIZATION AT PART LEVEL

Design parameterization must be carried out at both part and assembly levels. At the part level, sketch relations and dimensions must be defined to fully constrain the sketch profile of each solid feature and to capture the design intent. In addition, the geometry of the part will be regenerated following certain rules that were established when the part was created. In this section, the general modeling capabilities and the modeling procedure in CAD will be briefly reviewed. Guidelines for design parameterization at the part level will be presented, followed by examples.

5.4.1 PROFILE IN SKETCH

A general solid modeling procedure in CAD was discussed in Chapter 3. The solid modeling procedure usually starts with defining datum features, such as datum planes, datum coordinate systems, and datum axes, which serve as the references to facilitate solid feature creations. One of the datum planes is usually chosen to sketch a two-dimensional profile that is protruded to create the first (or base) solid feature.

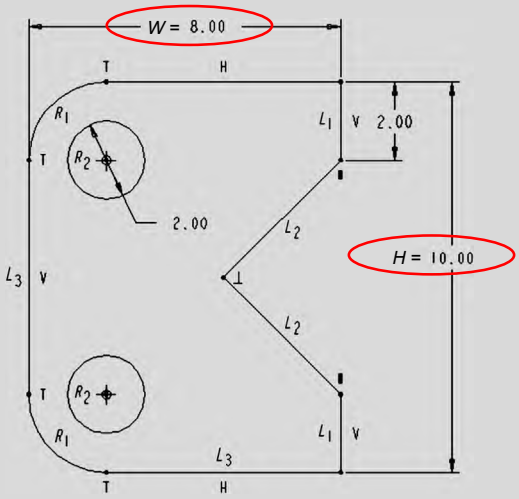
A build plan that describes the design intents, solid features, and sketch profiles with relations and dimensions is highly recommended. As discussed in Chapter 3, a build plan is especially useful for beginners to develop before creating any solid features.

In sketch, geometric entities, such as lines, arcs, and splines, are drawn as vectors for a single open or closed profile that can be protruded for a surface or solid feature or making a cut. A set of characteristic points is created for these vector entities. As discussed in Chapter 3, the profile is determined by the x- and y-positions of the characteristic points. In CAD, sketch relations, such as a concentric of circular arcs or parallel of lines, are usually generated automatically when these entities are drawn. The designer may define additional dimensions and sketch relations. Note that it is necessary to fully constrain the sketch profile in order to avoid unexpected errors while conducting design changes.

In general, the range of design variables is critical. When a profile dimension in a solid model is changed, the solid feature may become physically invalid. For a complex solid model, this problem may not be easy to detect. Determining the proper range of design variables that ensure valid solid features in advance is critically important for part parameterization. Note that finding the range for design changes is often conducted on a case-by-case basis. When more than one dimension is involved, the complexity of determining proper ranges for design changes is multiplied. Example 5.1 is presented to illustrate the point.

EXAMPLE 5.1

In the sketch profile shown in Figure 5.8(b), 11 characteristic points are created; therefore, 22 equations must be generated to determine their locations. As shown in the figure, geometric relations, including equal radii, concentric, perpendicular, and alignment, are imposed. As a result, four dimensions can be changed independently. In this example, we will focus on changing the height dimension (current value: $H = 10.00$), as circled in the figure below. What will be the width W of the sketch if the height dimension H is increased from 10 to 12? What are the upper and lower limits of the height dimension H?



Solutions

The first thing to ask is how the other entities will change when we vary the height dimension H. Because the equal length constraint L_3 is imposed on the bottom and left edges, changing H will affect the arc radius R_1 because the total width of the profile (current value $W = 8$) is retained.

In addition, the circles are concentric with the circular arcs; when the radius R_1 is changed, these two circles will move accordingly. Will the circles be pulled toward or pushed away from the arcs when the height dimension value increases? This question can be answered by the following equations.

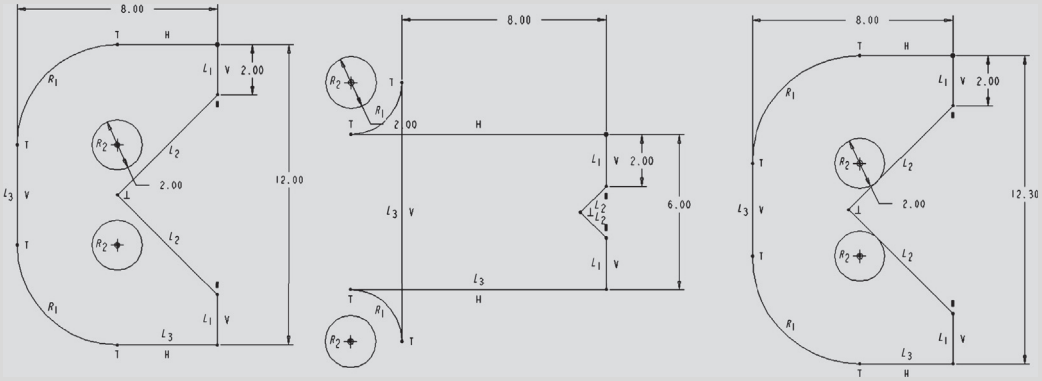
$$H = L_3 + 2R_1$$

$$W = L_3 + R_1$$

Hence, $R_1 = H - W$. Currently, H is 10 and W is 8; therefore, R_1 is 2. When H is increased from 10 to 12, R_1 becomes 4; that is, the circles are pushed away from the arcs, as shown in the figure below (Case A). Note that H cannot be set equal to W because it yields a zero-radius R_1, which is invalid and will not be accepted by CAD. When H is less than W, the radius will become negative; the profile will be regenerated as shown below (Case B), which is physically invalid. In addition, the height dimension cannot be too large. A large H value may cause the circles to run across the two 45° perpendicular edges, as shown in Case C. Therefore, the valid range of the height dimension is between 8 and 12.3, if W is kept unchanged.

Continued

EXAMPLE 5.1—cont'd



Case A Case B Case C

Note that the lower limit of H can be determined by $H = W + R_1$. Because W is 8 and R_1 is greater than 0, H must be greater than 8. How can the upper limit of the height H (which is 12.3) be determined? This is left as an exercise (hint: see Case C).

When the width dimension W is also considered for a change, the range of W depends on the current H value. If H is 12, the upper limit of W is 12. What will be the lower limit of W? (Again, this question is left as an exercise.) In general, defining the range of simultaneous changes for more than two design variables is not straightforward. Usually, all but one design variable is changeable.

5.4.2 SOLID FEATURES IN PART

When the first feature is protruded, parametric surfaces (Zeid, 1991; Mortenson, 2006) that represent the boundary of the solid feature are generated by CAD. After that, the designer may create additional datum, sketches, and protrusion features using options such as extrusion, sweep, revolve, and blend, as discussed in Chapter 3. The designer may also cut the existing features; generate chamfers or rounds; or copy, mirror, and/or pattern the existing features to create additional features.

When additional features are created, a model tree or feature tree is generated by CAD following the feature creation sequence. Boolean operations are employed to union or subtract the features from the previous ones according to their definitions. At the same time, the intersection curves between boundary surfaces are calculated to evaluate the Boolean operations and display the features. This is essentially the constructive solid geometry (CSG) method.

Note that in general the intersecting curves are approximated by interpolating a number of intersection points using B-spline or nonuniform rational B-spline curves (Zeid, 1991). The evaluated geometry and topology of boundary faces, edges, and vertices are stored in the CAD database for display. This is the boundary representation (B-rep) method. When features are being created, the designers can define relations in Pro/ENGINEER (or equations in SolidWorks) to relate feature dimensions to capture DIs. In this process, independent and dependent dimensions will be created to define a one-way relation. The independent dimensions become DVs. This is so-called *unidirectional*, *procedural*, or *parametric* modeling (McMahon and Browne, 1998).

Once all the features are created and relations are established, the part solid model is completely defined. When a design change is conducted by changing the DV values, the solid model will be regenerated in Pro/ENGINEER (or rebuilt in SolidWorks) by updating features (both datum and solid features) following the model tree, one at a time. Pro/ENGINEER or SolidWorks carries out steps (discussed in Chapter 3) to update individual features.

Note that modern CAD systems employ the concepts of both CSG and B-rep for solid modeling. In general, CSG keeps the relationship between features, whereas B-rep stores topological and geometric data for display and computations.

If the DIs are not properly captured in features and relations, the regeneration may lead to an undesirable or an invalid solid model. It is strongly recommended, especially for journeyman designers, that the designer creates a model build plan (with details of features, dimensions, and relations) before creating any features in CAD.

5.4.3 GUIDELINES FOR DESIGN PARAMETERIZATION

Based on the previous discussions, a set of guidelines for part parameterization is established in Table 5.1. These guidelines are separately listed according to the two axioms and the steps in solid modeling. Note that these guidelines are by no means complete. Readers may add more guidelines to suit their needs. These guidelines are not entirely objective. Some may be modified. The crankshaft example of the slider-crank mechanism is employed as an example to illustrate some of these guidelines.

The crankshaft is created in both Pro/ENGINEER and SolidWorks in the following sequence: three default datum planes (*DTM1*, 2, 3 or Front, Top, Right), a default coordinate system (CS0), a base extrusion feature, the lower extrusion feature, and the upper extrusion feature, as illustrated in Figure 5.9. Note that in both Pro/ENGINEER and SolidWorks, datum planes and coordinate systems are given for each part by default. They will be used as references for sketches and features (Guideline D1a). To simplify the presentation, only the Pro/ENGINEER model will be discussed. The steps in SolidWorks are similar.

The base extrusion feature is created by sketching its profile on *DTM3* (Guideline D1a) and extruding 0.5 unit along the normal direction of *DTM3*. The sketch is drawn using two semicircles and two straight-line segments, with the dotted lines (representing *DTM1* and *DTM2*) shown in Figure 5.9 as the references (Guidelines S1b). With the center points of the semicircles aligned with the references and various sketch relations (see Figure 5.10), only three dimensions are needed to completely define the sketch: the radii of the semicircles and the vertical distance between the center points (Guideline S2a, S2c, and S2d). Because the minimal number of dimensions is employed for the sketch, the crank length design variable can be easily captured in the base feature.

There are six characteristic points generated in the profile, as shown in Figure 5.10. Hence, it requires 12 independent equations to uniquely determine the positions of the characteristic points. This profile consists of six sketch relations, as listed in Figure 5.10, and three dimensions. These 12 equations can be formulated by employing the sketch relations and dimensions, as shown in Figure 5.10. Note that, in this case, these equations are linear; hence, they can be solved by matrix operations. When a design variable is changed in the sketch, the same set of equations is solved for the new positions, hence updating the sketch profile.

Table 5.1 Guidelines for Part Parameterization

	Independence Axiom (1)	Information Axiom (2)
Datum (D)	D1a: A solid model should always start with the default datum features (i.e., three orthogonal datum planes) and the default coordinate system D1b: Additional datum features should be referenced to the default datum features instead of geometric entities (e.g., an edge) of a solid feature whenever possible	D2a: Never duplicate datum features
Sketch (S)	S1a: A sketch should be created on a default datum plane if possible (instead of on a face of an existing solid feature) in order to minimize parent−child coupling between solid features S1b: Dimensions on a sketch should be created by using datum features as references instead of geometric entities (e.g., an edge) of a solid feature whenever possible S1c: A design variable (DV) should never be referenced to another geometric dimension (unless the design intent requires it)	S2a: One characteristic point of the sketch profile should be anchored to default datum features (e.g., intersection of two datum planes) S2b: A face and a geometric entity of an existing feature can be chosen as the sketch plane and the anchor point, respectively, only for the purpose of capturing a design intent (DI) S2c: Geometric entities on a sketch profile should be aligned to datum features or existing entities to minimize the number of dimensions S2d: Sketch relations should be defined as much as possible to reduce the number of dimensions S2e: The information contents should be minimized by using symmetry constraint S2f: Relations between dimensions must be added not only to minimize the information content but also to capture the DI. It is desirable to define fewer relations by adding more sketch relations S2g: Redundant and zero-valued dimensions should never be defined S2h: Range of the design variables should be determined in advance
Solid features (F)	F1a: A solid feature should be decoupled from existing solid features by referencing only to default datum features whenever possible	F2a: Use attribute of the solid feature instead of addition dimension to define the feature; for example, a through hole should not be created with an extrusion cut with a depth dimension of larger value F2b: The amount of information in solid features that have one or more planes of symmetry can be minimized using pattern, copy, and mirror
Parts (P)	P1a: After the solid model is built, the designer should only have access to dimensions that form a DI (i.e., only to the DVs) P1b: Relate dimensions directly to the DV to avoid loop or chain relations	P2a: Define relations between dimensions of different features to capture the design intent

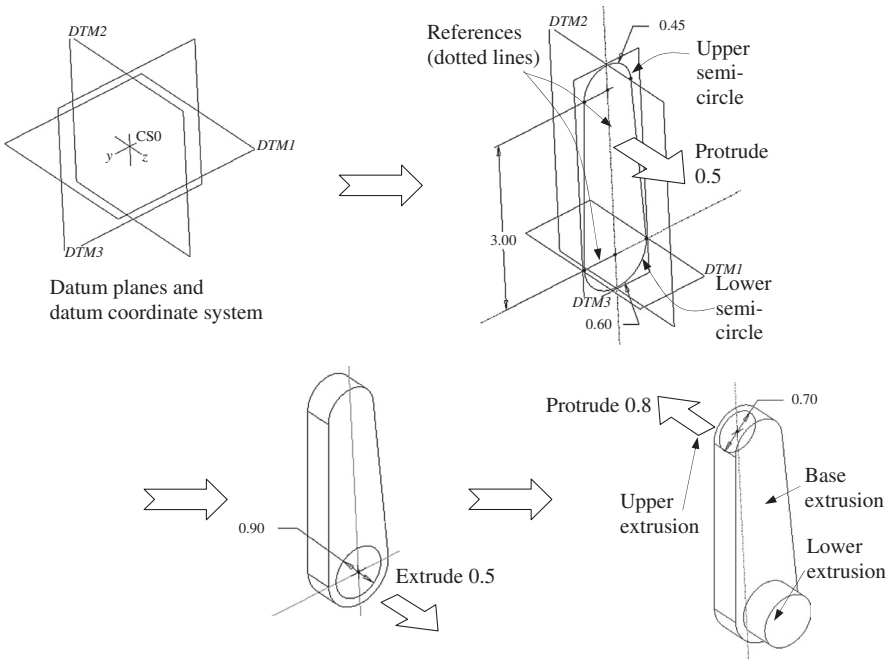
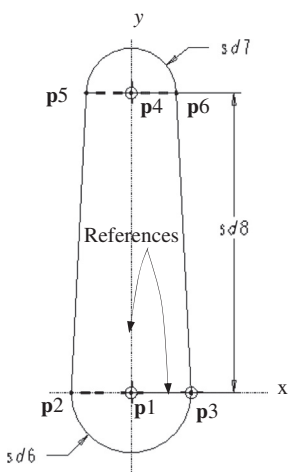


FIGURE 5.9

Feature creation steps of the crankshaft.



Implicit constraints or sketch relations:
C1: **p**1 aligned with references (*x*- and *y*-axes);
C2: **p**3 aligned with reference (*x*-axes);
C3: **p**4 aligned with reference (*y*-axes);
C4: **p**1 and **p**2 aligned horizontally;
C5: **p**4 and **p**5 aligned horizontally;
C6: **p**4 and **p**6 aligned horizontally.

Variational equations:

$p1x = 0$	(C1);	$p4x - p1x = 0$	(C3);
$p1y = 0$	(C1);	$p4y - p1y = sd8;$	
$p2y - p1y = 0$	(C4);	$p5y - p4y = 0$	(C5);
$p1x - p2x = sd6;$		$p4x - p5x = sd7;$	
$p3x - p1x = sd6;$		$p6x - p4x = sd7;$	
$p3y - p1y = 0$	(C2);	$p6y - p4y = 0$	(C6).

FIGURE 5.10

Variational equations for the sketch profile.

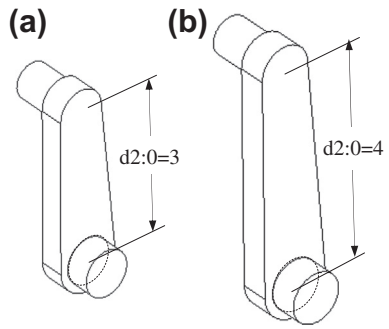


FIGURE 5.11

Design intent captured as $d2{:}0 = 3$ (a) and $d2{:}0 = 4$ (b).

As shown in Figure 5.9, the lower extrusion feature is created by sketching a circle of diameter 0.9 that is concentric with that of the lower semicircle of the base feature (Guideline S2b) and extruding 0.5. Similarly, the upper extrusion feature is created by sketching a circle of diameter 0.7 that is concentric with that of the upper semicircle of the base feature (Guideline S2b) and extruding 0.8 in the opposite direction.

By imposing the alignment and concentric rules, the crankshaft is properly parameterized, yet the number of dimensions in the crankshaft solid model is minimized. The change of crank length can be realized by simply modifying the dimension $d2{:}0$, as shown in Figure 5.11. The base extrusion feature is updated according to its sketch shown in Figure 5.10. The lower extrusion feature is unchanged because its sketch profile is concentric with the lower semicircle of the base feature. The upper extrusion feature is pushed upward because its center point coincides with that of the upper semicircle of the base feature and the center point moves up due to the references chosen. The change is propagated to features in the crankshaft through a model tree (or a CSG tree) established following the feature creation sequence and Boolean operations, as shown in Figure 5.12.

5.5 DESIGN PARAMETERIZATION AT ASSEMBLY LEVEL

Before reading this section, you are encouraged to review Chapter 4 for the mating constraints employed in CAD that support assembly modeling. With the understanding of the mating constraints, we discuss the guidelines for assembly developed following the two axioms. The slider-crank example is used to illustrate the assembly capabilities and design parameterization in both Pro/ENGINEER and SolidWorks.

5.5.1 GUIDELINES FOR DESIGN PARAMETERIZATION

Similar to the part level, the DIs in assembly can be uncoupled, coupled, and decoupled. The uncoupled design is again always superior to the others, according to the independent axiom. However, uncoupled DIs may not always be possible in practical applications. In general, it is required that the

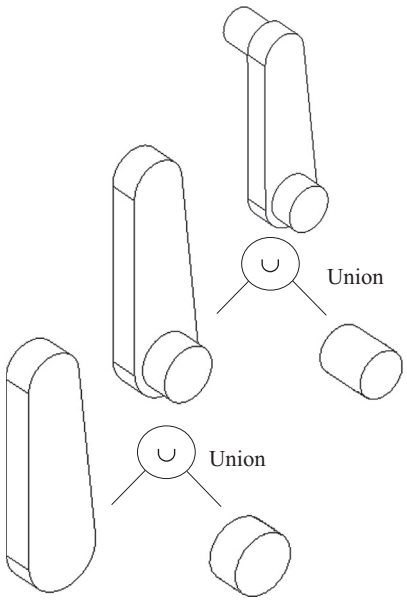


FIGURE 5.12

The constructive solid geometry tree.

designer decouple the coupled DIs by adding DVs that alleviate the coupling effect, as discussed previously.

The information axiom at the assembly level can be exercised by adding relations or equations for dimensions across parts. For example, the diameter of a shaft must be related to that of the hole it inserts into to reduce the number of DVs and capture the DIs. Note that, at the assembly level, in addition to complying with the two axioms, mating constraints and datum features must be properly defined to capture DIs.

A set of guidelines for assembly is listed in Table 5.2. Again, more guidelines may be added, and some of the guidelines stated in Table 5.2 may be subjective. The slider-crank assembly is employed to illustrate these guidelines.

In addition to the guidelines stated above, the following are useful tips:

- Fix mate errors as soon as they occur. Adding mates never fixes earlier mate problems.
- If a component is causing problems, it is often easier to delete all its mates and recreate them instead of diagnosing each one.
- Whenever possible, fully define the position of each part in the assembly, unless you need the part to move to visualize the assembly motion.
- Assemblies with many available degrees of freedom (DOF) take longer to solve, and have less predictable behavior when you drag parts. Drag components to check their remaining degrees of freedom.

Table 5.2 Guidelines for Assembly Parameterization

	Independence Axiom (1)	Information Axiom (2)
Datum (D)	D1a: An assembly should always start with the default datum features (i.e., three orthogonal datum planes) and the default coordinate system D1b: Additional datum features should be referenced to the default datum features whenever possible	D2a: Never duplicate datum features
Mating constraints (C)	C1a: Whenever possible, mate all components to one or two fixed components or references Long chains of components take longer to solve and are more prone to mate errors C1b: Do not create loops of mates. They lead to mate conflicts when you add subsequent mates C1c: Drag components to test their available degrees of freedom and see if the design intent is captured	C2a: Avoid redundant mates. Although CAD allows some redundant mates, these mates take longer to solve and make the mating scheme harder to understand and diagnose if problems occur C2b: Eliminate all degrees of freedom (DOF), except for those needed for kinematic analysis
Assembly dimensions (A)	A1a: Define relations across parts to capture DI	A2a: Minimize the number of dimensions and relations while assembling parts. The assembly options that require defining more dimensions and relations should only be used when the new dimension is a DV

5.5.2 SLIDER-CRANK ASSEMBLY IN PRO/ENGINEER

At the assembly level, the intent is to orient the crankshaft vertically and align the piston and piston pin horizontally with the center point of the lower shaft of the crankshaft, as shown in Figures 5.13(a) and (b). Three assembly datum planes and a datum coordinate system are given by default. The crankshaft is assembled by properly aligning its datum planes with the assembly datum planes for a vertical orientation, as shown in Figure 5.13(a) (Guideline D1a). In order to assemble the rod, two additional datum planes are created in the assembly. *ADTM4* is created by offsetting *ADTM2* 3 units upward, as shown in Figure 5.13(b) (Guideline D1b). The datum plane *ADTM5* is created by rotating *ADTM4* with an angle $d1{:}1 = \sin^{-1}(3/8)$. Note that *ADTM5* will be used to orient the rod (Guideline D1b). The rod is assembled to the crankshaft by three mating constraints: axis alignment, surface mate, and surface alignment, as shown in Figure 5.14.

In addition, the vertical position of *ADTM4* and the rotation angle of *ADTM5*, which determine the configuration of the assembly, will be related to the crankshaft and rod lengths through the following equations:

$$d0{:}1 = d2{:}0 \tag{5.2}$$

$$d1{:}1 = \sin^{-1}\left(d2{:}0/d3{:}2\right). \tag{5.3}$$

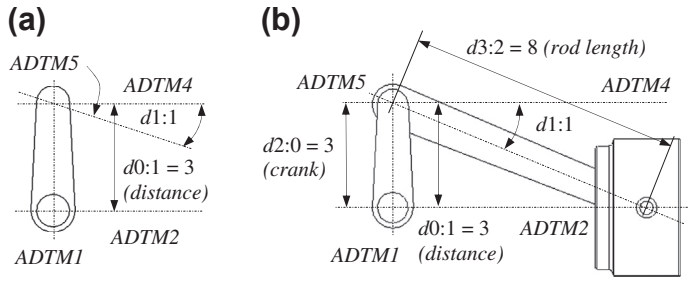


FIGURE 5.13

Parameter relations. (a) Crankshaft. (b) Assembled slider-crank mechanism.

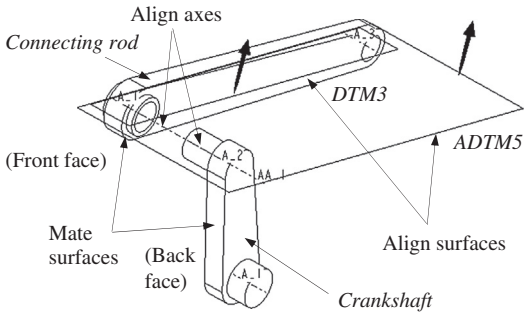


FIGURE 5.14

Assembly mating constraints defined for rod.

Note that Eq. 5.2 defines a relation that moves the datum plane *ADTM4* up or down according to the crank length (*d*2:0). Eq. 5.3 defines the trigonometric relation of angle *d*1:1 to the design variables *d*2:0 (crank length) and d*3*:2 (rod length). Dimension *d*1:1 actually rotates *ADTM5* according to the changes of *d*2:0 and *d*3:2. This is how the slider-crank mechanism is parameterized. These equations define two independent design variables (i.e., *d*2:0 and *d*3:2) by relating four dimensions in assembly. Therefore, the information contents of the first design intent are minimized (Guideline A2a). Details about this assembly are available in the tutorial lesson P1.3.

Note that the way that the slider-crank mechanism is parameterized, as discussed above, is not unique. They are presented for the purpose of illustrating some of the guidelines listed in Table 5.2. Other ways of parameterizing this mechanism exist. For example, instead of offsetting *ADTM2* for *ADTM4* with a dimension *d*0:1, *ADTM5* can be created by rotating *ADTM1* along the datum axis A_2 of the crankshaft (see Figure 5.14 for axis A_2). By doing so, dimension *d*0:1 can be removed, and Eq. 5.2 is not necessary, thus further reducing the information content.

5.5.3 SLIDER-CRANK ASSEMBLY IN SOLIDWORKS

The slider-crank mechanism is assembled in SolidWorks in a slightly different way. Because one of the objectives in SolidWorks assembly is to conduct kinematics analysis of the mechanism, as illustrated in Figure 5.15(a), a bearing part is introduced and is fixed in the assembly, as shown in Figure 5.15(b). Moreover, no additional datum plane is needed to orient the rod because its orientation will be determined by SolidWorks when the crankshaft rotates.

The crankshaft is assembled to bearing using *Concentric* and *Coincident* constraints, leaving one rotational DOF (please refer to Figure 4.7a in Chapter 4 for details). The connecting rod and piston pin are assembled in a similar way, also leaving one rotational DOF for each part assembled (please see Figure 4.7b). The piston is assembled using one *Concentric* and two *Coincident* constraints, as shown in Figure 5.16. Note that the second *Coincident* constraint that coincides with *Plane3* of the piston and *Plane2* (horizontal plane) of the bearing confines the movement of the piston horizontally. When the length of the crankshaft or rod is changed, the assembly will be rebuilt, as shown in Figure 5.17,

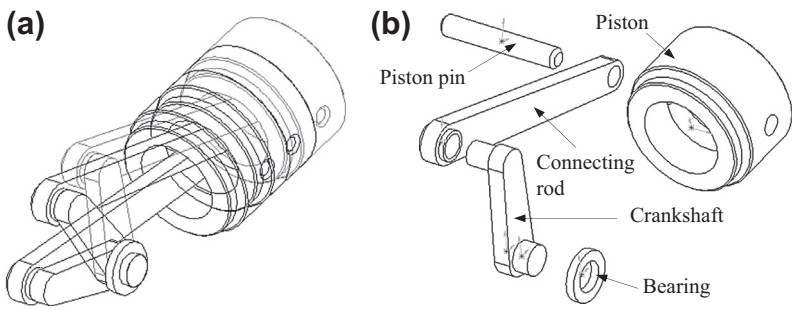


FIGURE 5.15

Slider-crank assembly in SolidWorks. (a) Kinematic analysis. (b) Exploded view.

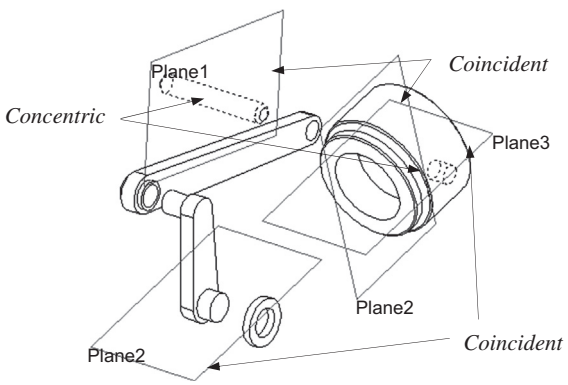


FIGURE 5.16

Mating constraints defined for the piston.

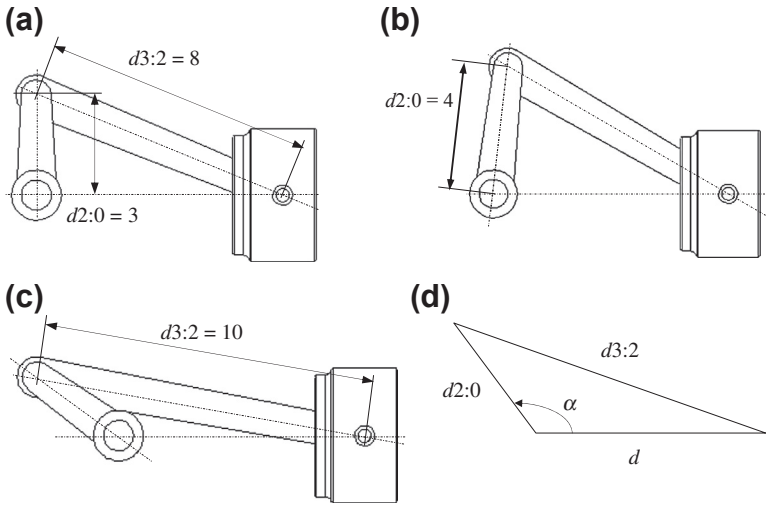


FIGURE 5.17

Change of length design variables in SolidWorks. (a) Design variables $d2{:}0$ and $d3{:}2$. (b) $d2{:}0$ changes to 4. (c) $d3{:}2$ changes to 10. (d) Trigonometric relation of the assembly dimensions.

according to the trigonometric equation (see Figure 5.17(d)), with the distance d between the piston and the crankshaft fixed temporarily:

$$\alpha = \sin^{-1}\left(\frac{d2{:}0^2 + d^2 - d3{:}2^2}{2 \times d2{:}0 \times d}\right). \tag{5.4}$$

5.6 CASE STUDIES

Two case studies are presented to demonstrate the parameterization method for practical applications, including a single-piston airplane engine and a HMMWV suspension. Parameterization at both part and assembly levels is discussed.

5.6.1 SINGLE-PISTON ENGINE

Solid models of a single-piston airplane engine were created in Pro/ENGINEER, as shown in Figures 5.1 and 5.18. The models, consisting of 18 parts, were first created by a third party without adequate parameterization. Even though this original model is geometrically valid, it is not properly parameterized. As a result, any simple change will lead to invalid features, parts, and hence assembly.

Moreover, the amount of information the model carries is sometimes excessive or, in other cases, incomplete. A typical example of the use of excessive information is shown in Figure 5.19(a), in which redundant dimensions are founded. For instance, $d45$ and $d22$ both specify the depth of the inner hole of the horizontal cylinder of the case, and $\phi d44$ and $\phi d21$ refer to the diameter of the same hole. In

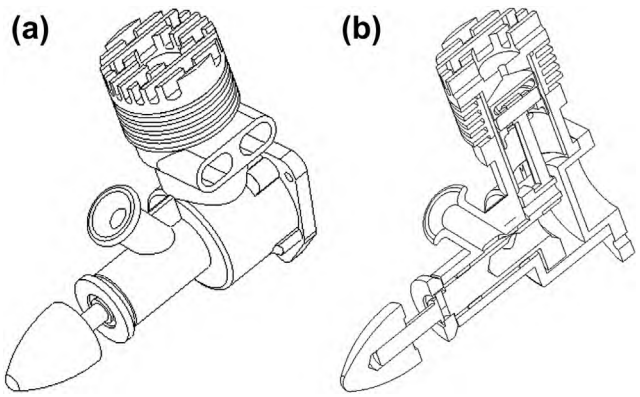


FIGURE 5.18

The single-piston engine. (a) Unexploded view. (b) Section view.

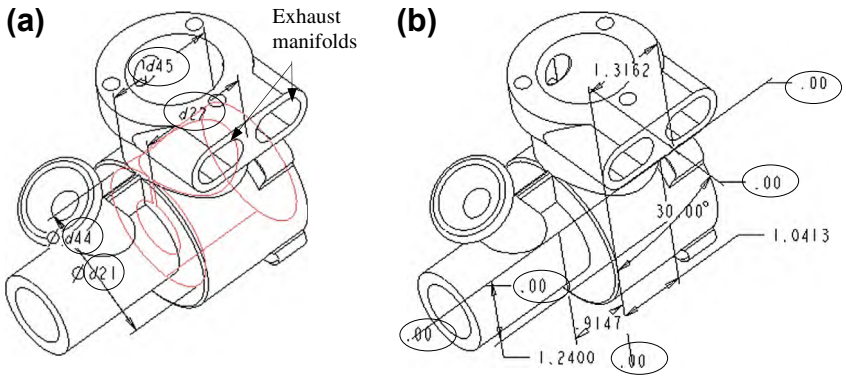


FIGURE 5.19

Excessive information in the original model. (a) Redundant dimensions. (b) Dimensions with zero value.

addition, as shown in Figure 5.19(b), there are several dimensions with zero value as circled, which must be removed.

The DIs of the engine design are defined as:

DI1: The stroke length, composed of the crankshaft length $d6{:}10$ (10 represents crankshaft in engine assembly), as shown in Figure 5.20(a).
DI2: The volume above the piston at top dead center position, composed of the bore diameter d46: 0, and the length of the connecting rod $d0{:}14$, as shown in Figures 5.20(b) and (c), respectively.

Modifications to all 18 parts in the original model were carried out in order to comply with Axioms 1 and 2 at both the part and assembly levels. A typical example, the engine case, is presented to illustrate how the guidelines are followed at the part level. Note that changes in any of the three DVs

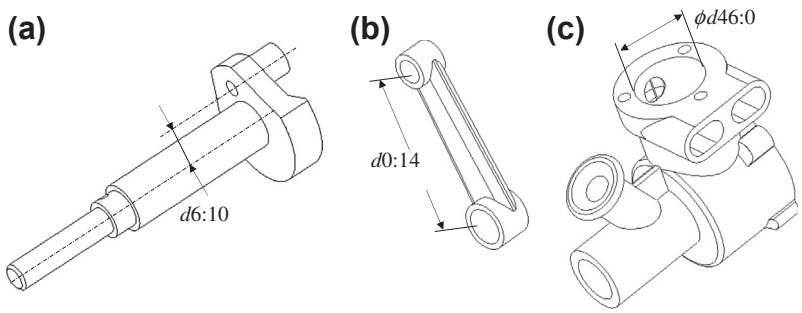


FIGURE 5.20

Design variables defined for the DIs. (a) Crankshaft. (b) Connecting rod. (c) Engine case.

will affect the engine case. The objective is to parameterize the case so that $d46{:}0$ can be changed independently, and changes in $d6{:}10$ and $d0{:}14$ will propagate to the engine case correctly.

5.6.1.1 Part Level: Engine Case

Independence Axiom: Three default orthogonal datum planes are kept in the case solid model (Guideline D1a). Redundant datum planes that coincide with the three default datum planes are deleted (Guideline D2a). For example, *DTM1* and *DTM4* coincide with each other, and *DTM4* is deleted. Before deleting *DTM4*, all the features that were referenced to *DTM4* must be redefined by referring them to *DTM1* (Guideline D2b).

Also, a design variable should be referred to nonchanging features, such as datum planes (Guideline S1c). In the engine case, the bore is created as a hole with its axis intersected by *DTM1* and *DTM3*, as shown in Figure 5.21(a). Consequently, the design variable $d46{:}0$ is not dependent on other dimensions, but refers to the nonchanging datum features.

The dimensions that are not DVs should not be available to the designer. However, they should be updated automatically via relations (Guideline P2a). Neither $d21$ nor $d22$ are DVs. The depth of the

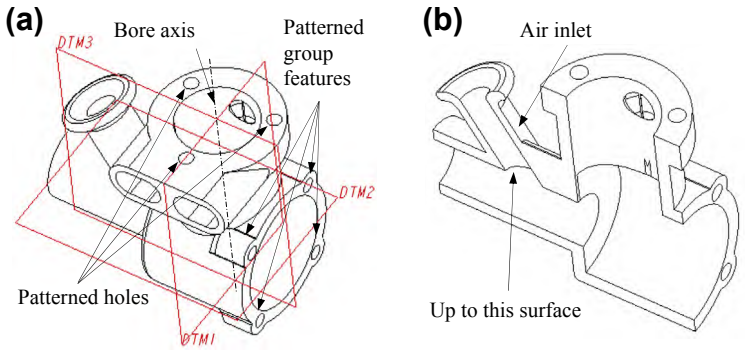


FIGURE 5.21

Engine case. (a) Datum planes. (b) Air inlet (section view).

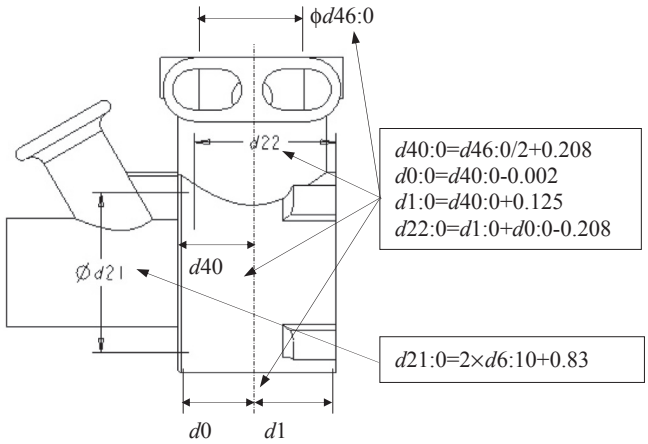


FIGURE 5.22

Relations defined in the engine case.

hole *d*22:0 is related to the bore diameter *d*46:0, and the diameter of the hole *d*21:0 depends on the length of the crankshaft *d*6:10. The relations and the modified solid model of the case are shown in Figure 5.22.

Information Axiom: Redundant and zero-value dimensions must be eliminated (Guideline S2g). In the engine case, both the depth and the diameter of the hole were redundantly defined in the original model, as shown in Figure 5.19(a). Even though *d*44 and *d*21 have the exact same value, they belong to two different features. Dimension *d*44 is eliminated by aligning the circle defined by *d*44 to the circle defined by *d*21 (Guideline S2c). A similar problem arises for dimensions *d*45 and *d*22. Dimension *d*45 is eliminated by resorting to alignment (Guideline S2c). In addition, all the zero-value dimensions are removed because they do not compose any of the DIs (Figure 5.19(b)).

The air inlet is defined as a through hole, instead of a hole with depth up-to-surface (Guideline F2a). Consequently, it will always be a through hole when a design change is committed. Also, copy and mirror are used in the creation of one of the exhaust manifolds (Guideline F2b), as shown in Figure 5.21(a). It is desired that both manifolds maintain the same dimensions. In the original model, they were defined independently. In this case, one design change must be implemented twice, which is unnecessary and error-prone. The same guideline is applied to pattern the three holes on top of the case (see Figure 5.21(a)). In addition, the group of extrusion, round, and hole features, as shown in Figure 5.21(a), is patterned for an additional three instances. Furthermore, the dimensions are properly related (see Figure 5.22) to capture the DIs (Guideline P1a).

5.6.1.2 Assembly Level: Engine

The other 17 parts are also parameterized following the guidelines at part level for the two DIs. At the assembly level, guidelines are followed for assembling the 18 parts.

Independent Axiom: Default assembly datum features, including the three orthogonal datum planes and coordinate system, are used as references (Guideline D1a). Proper mating constraints are employed to assemble all 18 parts without looping (Guideline C1b).

Information Axiom: Only two rotational DOF (between connecting rod and piston pin and between crankshaft and connecting rod) are kept for kinematic analysis. All the other DOF are eliminated (Guideline C2b). Dimensions are related across the parts (e.g., *d*21:0 in Figure 5.22) to minimize the contents of the DIs (Guideline A2a). Moreover, relations are created for the length of the piston pin, diameter of the piston, and the bore diameter of the case. The DIs are properly captured at the assembly level, as illustrated in Figure 5.1.

5.6.2 HMMWV SUSPENSION

The HMMWV solid model, discussed in Chapter 1, was initially created in Pro/ENGINEER and then converted to SolidWorks, partially for testing the process of solid model conversion between CAD systems. There are more than 200 parts and assemblies (see Figure 5.23(a)). The suspension is modeled in detail (Figure 5.23(b)) such that the main characteristics of the vehicle performance can be captured accurately in motion simulation. A more detailed view of the front right suspension quarter is shown in Figure 5.24(a).

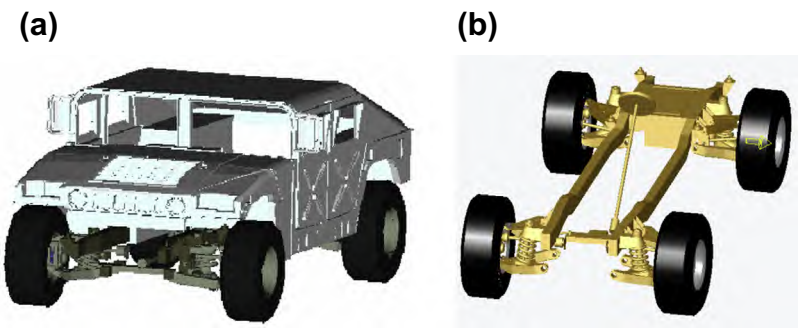


FIGURE 5.23

HMMWV CAD model. (a) Vehicle assembly. (b) Suspension assembly.

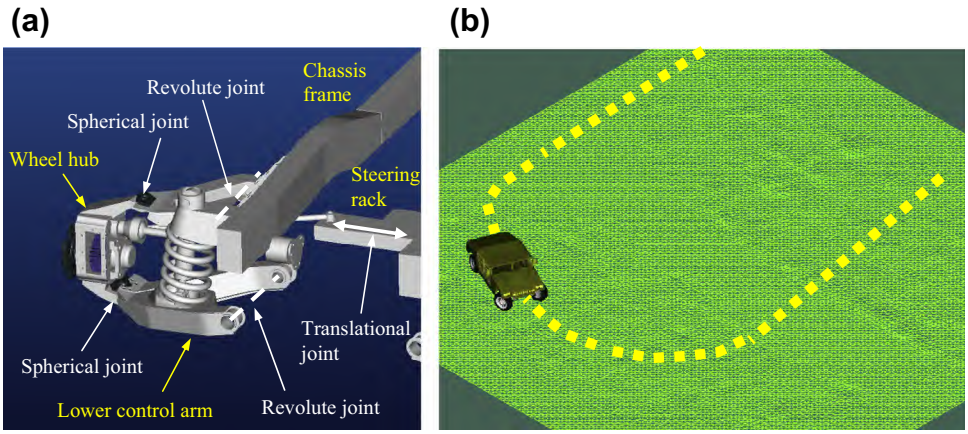


FIGURE 5.24

HMMWV dynamic simulation model. (a) Front-right suspension. (b) Vehicle motion simulation.

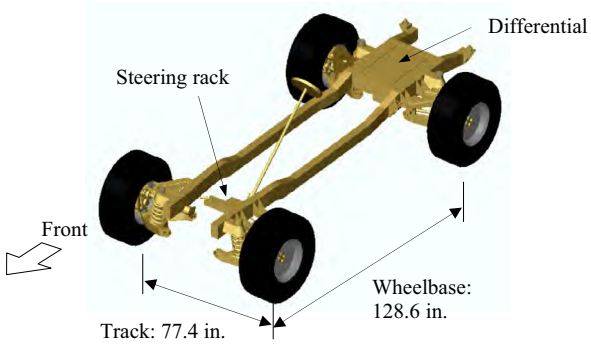


FIGURE 5.25

Track and wheel base design variables.

A dynamic simulation model was created for dynamic simulation on a bumpy 100 ft × 100 ft terrain, as shown in see Figure 5.24(b). The vehicle vibrates significantly towards the later stages of the simulation due to the bumpy road conditions. The overall design goal was to optimize the vehicle dynamic characteristics (Chang and Joo, 2006).

The vehicle track and wheelbase shown in Figure 5.25 are the two primary design variables defined for HMMWV suspension. In order to support HMMWV design optimization, the suspension assembly must be parameterized in CAD (in this case, SolidWorks). The design parameterization must be conducted at both part and assembly levels.

5.6.2.1 Track Design Variable

For the track design variable, two parts are involved: differential (Figure 5.26(a)) and steering rack (Figure 5.26(b)). The geometry of both parts is simple, and their width dimensions are to be related to capture the track design variable. The outer width of the differential $d2$@sketch1, as shown in Figure 5.26(a), is chosen as an independent dimension. All the geometric features in the differential will be changed according to $d2$@sketch1 following the relations defined in Table 5.3. The relations show that $d1$@sketch2, $d1$@sketch3, and $d1$@sketch4 will be changed according to $d2$@sketch1. In addition, $d2$@sketch3, $d2$@sketch2, and $d2$@sketch4 are fixed. Note that in the equations of Table 5.3, dimensions shown on the left-hand side of the equal sign become dependent.

For the steering rack shown in Figure 5.26(b), dimension $d1$@sketch1 is chosen as independent, and $d6$@sketch1 will be changed with the same amount as $d1$@sketch1, as defined by the first equation listed in Table 5.4. Dimensions $d1$@sketch10 and $d6$@sketch10 are related to $d1$@sketch1 and $d6$@sketch1 via the last two equations shown in Table 5.4, respectively, with a fixed wall thickness of $d3$@sketch10 = 0.53033 in.

At the assembly level, mating constraints are defined for the differential and both frame rails, as shown in Figure 5.27(a). First, the side faces of the differential and frame are assembled using surface coincident (mate) constraints. In addition to surface coincident constraints, point coincident constraints are added between the corner points of the differential and points on the top edge of the frame rails. The steering rack is assembled to the tie-rod on each side by using concentric (axis alignment) and surface coincident (mate) constraints, as shown in Figure 5.27(b).

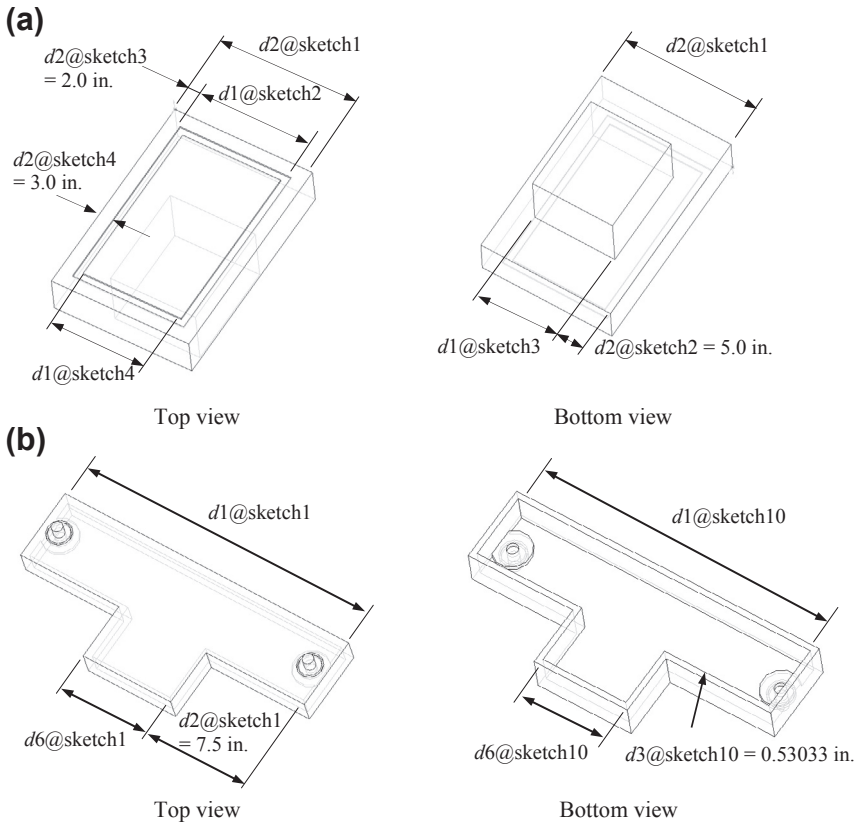


FIGURE 5.26

Design parameterization for track design variable at the part level. (a) Design parameterization for the differential. (b) Design parameterization for the steering rack.

Table 5.3 Relations Defined for the Differential

Equations	Design Intents
$d1$@sketch2 = $d2$@sketch1−2×$d2$@sketch3	$d2$@sketch1 is independent, $d2$@sketch3 = 2.0, and is fixed
$d1$@sketch3 = $d2$@sketch1−2×$d2$@sketch2	$d2$@sketch1 is independent, $d2$@sketch2 = 5.0, and is fixed
$d1$@sketch4 = $d2$@sketch1−2×$d2$@sketch4	$d2$@sketch1 is independent, $d2$@sketch4 = 3.0, and is fixed

Table 5.4 Relations Defined for the Steering Rack

Equations	Design Intents
$d6$@sketch1 = $d1$@sketch1−2×$d2$@sketch1	$d1$@sketch1 is independent, $d2$@sketch1=7.5 in., and is fixed
$d1$@sketch10 = $d1$@sketch1−2×$d3$@sketch10	$d1$@sketch1 is independent, wall thickness $d3$@sketch10 = 0.53033 in. fixed
$d6$@sketch10 = $d6$@sketch1−2×$d3$@sketch10	Wall thickness $d3$@sketch10 = 0.53033 in. fixed

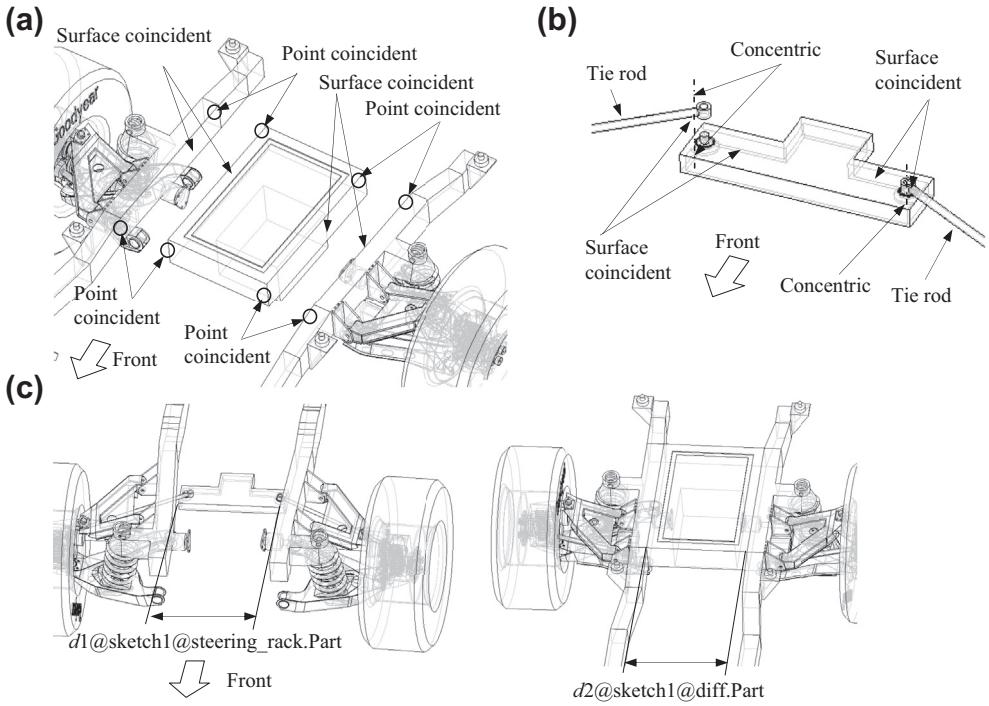


FIGURE 5.27

Design parameterization for track design variable at the assembly level. (a) Mating constraints defined between the differential and frame. (b) Mating constraints defined between steering rack and tie-rods. (c) Relation between widths of the differential and steering rack.

Next, the relationship between the width of the differential and width of the steering rack is defined at assembly level, as shown in Figure 5.27(c). The relationship between dimensions $d1$@sketch1 in the steering rack and $d2$@sketch1 in the differential is defined as $d1$@sketch1@steering_rack.Part = $d2$@sketch1@diff.Part, so that widths of the steering rack and differential change simultaneously. Therefore, $d2$@sketch1@diff.Part represents the track design variable which is independent. Note that $d1$@sketch1@steering_rack.Part and $d2$@sketch1@diff.Part have the same numerical value.

5.6.2.2 Wheelbase Design Variable

Defining the wheelbase design variable is straightforward. It involves changing the length of the two center frame rails at the same time (Figure 5.28(a)). The center frame rails are assembled to the rear frame using surface coincident constraints as well as point coincident constraints at the end faces of the frame (Figure 5.28(b)). Similar constraints are defined for assembling the center frame rails to the front frame. A relation $d1$@sketch5@right_frame.Part = $d1$@sketch3@left_frame.Part is defined to capture the wheelbase design variable represented by $d1$@sketch3@left_frame.Part. As a result, when $d1$@sketch3@left_frame.Part is increased, the rear portion of the vehicle gets pushed backwards, and vice versa. Note that when the track or wheelbase design variable is changed, both mass properties and

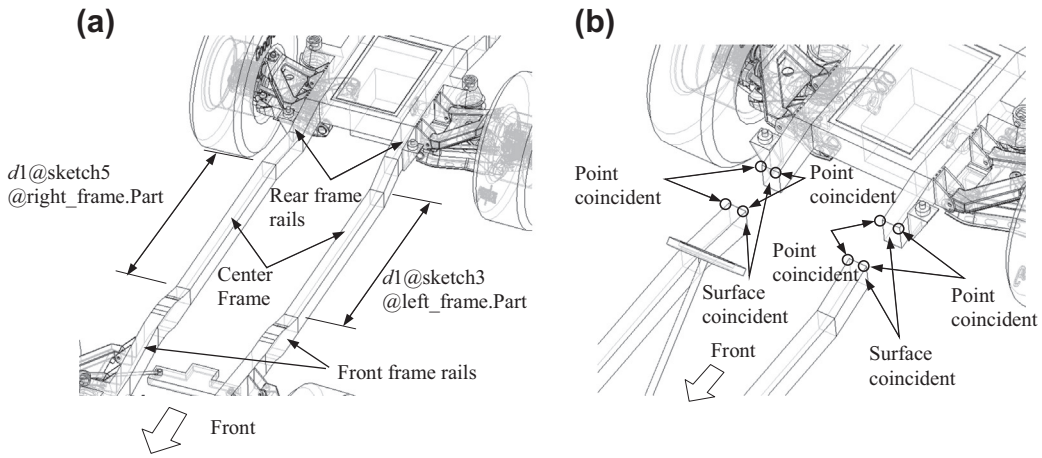


FIGURE 5.28

Design parameterization for the wheelbase design variable. (a) Relation between two center frame rails. (b) Mating constraints defined for the center and rear frame.

joint locations of the HMMWV vehicle model are altered, therefore varying the vehicle dynamic performance. In this case, both the track and wheelbase design variables are called global design variables.

5.6.2.3 Design Change

The change of HMMWV suspension geometry due to the change of these two design variables is shown in Figure 5.29. The suspension is properly parameterized. The parameterized HMMWV model was employed to search for a design that optimizes the dynamic characteristics of the suspension (Chang and Joo, 2006).

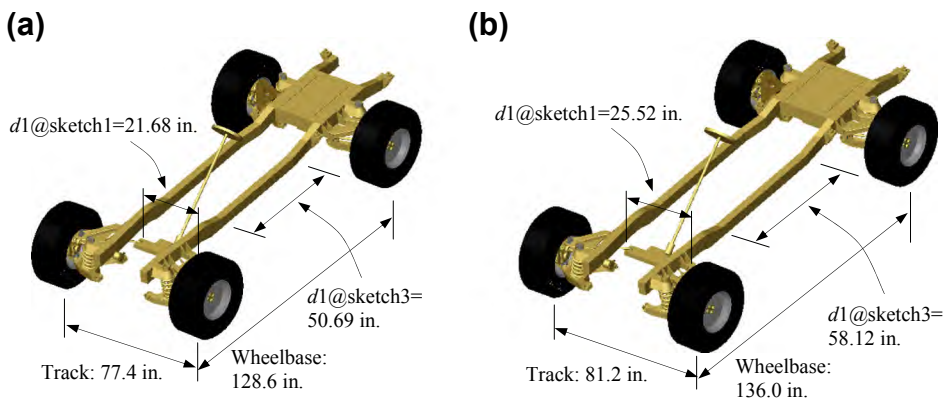


FIGURE 5.29

HMMWV suspension at initial design (a) and optimal design (b).

5.7 SUMMARY

In this chapter, solid modeling and assembly techniques implemented in Pro/ENGINEER and SolidWorks were discussed. Usually in e-Design, a design change is first realized by modifying geometric dimension values in CAD and automatically regenerating or rebuilding the solid models. To capture a DI in CAD, the product solid model must be properly created and parameterized. With this understanding, a design parameterization method that supports the capture of design intents has been introduced.

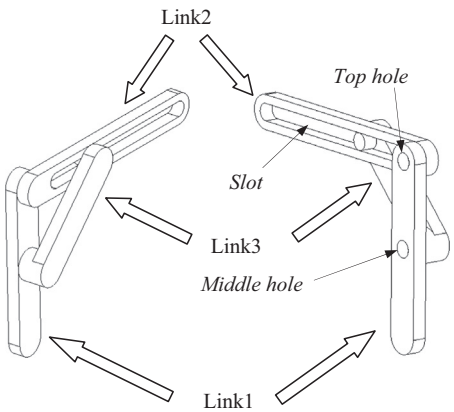
Design intent for a single part can be captured by properly creating individual solid features and carefully relating dimensions within or among features. Consequently, when a dimension value is changed, the solid features affected by the change can be regenerated or rebuilt successfully. At the assembly level, DI is captured by defining adequate mating constraints and relating dimensions across parts so that a change in dimension value can be propagated to all parts affected. The parts affected must be regenerated successfully; at the same time; they must maintain proper positions and orientations with respect to one another without violating any assembly mating constraints nor revealing part penetration or excessive gaps. Moreover, the regenerated solid model must meet the designer's expectations.

Design parameterization guidelines based on the independent and information axioms have been introduced. These guidelines will facilitate the creation of parametric solid models that support design engineers in exploring design alternatives in the e-Design environment.

QUESTIONS AND EXERCISES

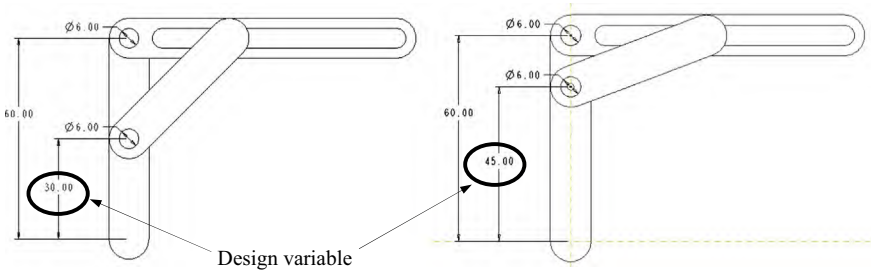
5.1. Show how the upper limit of the height dimension $H = 12.3$ in Example 5.1 is determined based on the geometry of the sketch profile. If H is set to 12, determine the upper and lower limits of the width design variable W.

5.2. Use the given three parts (link1, link2, and link3) to create an assembly like the one shown below.



Required configuration:

- Link1 must be vertical;
- Link2 must be horizontal, and the shaft on link2 must insert to the top hole in link1.
- The two shafts of link3 must insert to the middle hole of link1 and slot of link2.
  - **a.** Show/explain what mating constraints you employed for this assembly.
  - **b.** Define the vertical location of the middle hole of link1 as the design variable, as shown in the next figure, and find the following:



Upper bound of the design variable without interference (showing how you reached the answer):

_____

Lower bound of the design variable without interference (showing how you reached the answer):

_____

REFERENCES

Chang, K.H., Joo, S.-H., July 2006. Design parameterization and tool integration for CAD-based mechanism optimization. Advances in Engineering Software 37, 779–796.

Chang, K.H., Silva, J., Bryant, I., December 1999. Concurrent design and manufacturing for mechanical systems. Concurrent Engineering Research and Applications (CERA) Journal 7 (4), 290–308.

Lee, K., 1999. Principles of CAD/CAM/CAE Systems. Addison Wesley Longman, Inc, ISBN: 0-201-38036-6.

McMahon, C., Browne, J., 1998. CADCAM, second ed. Addison-Wesley, ISBN: 0-201-17819-2.

Mortenson, M.E., 2006. Geometric Modeling, third ed. Industrial Press.

Suh, N.P., 1990. The Principles of Design, Oxford Series on Advanced Manufacturing. Oxford University Press, New York, NY.

Toogood, R., Zecher, J., 2012. Creo Parametric 1.0 Tutorial and MultiMedia DVD. SDC Publication.

Zeid, I., 1991. CAD/CAM Theory and Practice. McGraw-Hill, Inc, ISBN: 0-07-072857-7.

CHAPTER

PRODUCT DATA MANAGEMENT 6

CHAPTER OUTLINE

6.1 Introduction 267
6.2 File Management 269
6.2.1 AD-HOC Methods 270
6.2.2 PDM Approach 273
6.3 Fundamentals of PDM 274
6.3.1 Engineering Data Models 275
6.3.1.1 Product Data Model 275
6.3.1.2 Process Data Model 279
6.3.2 Basic Functions of PDM Systems 280
6.3.2.1 User Functions 281
6.3.2.2 Utility Functions 282
6.3.3 Benefits of PDM Systems 283
6.3.4 Impact to Industry 284
6.4 PDM Systems 285
6.4.1 Systems Offered by CAD Vendors 286
6.4.1.1 AutoDesk® ProductStream® of Autodesk Inventor 286
6.4.1.2 ENOVIA Smarteam of CATIA 286
6.4.1.3 Windchill by PTC 287
6.4.1.4 TeamCenter by Siemens UGS NX 287
6.4.1.5 SolidWorks Enterprise PDM 288
6.4.2 Systems Offered by Non-CAD Vendors 288
6.4.2.1 SofTech ProductCenter® PLM 288
6.4.2.2 Arena Cloud PLM 288
6.5 Product Data Exchange 289
6.5.1 Data Exchange Options 291
6.5.2 Direct Model Translations 292
6.5.2.1 Importing Pro/ENGINEER Parts to SolidWorks 292
6.5.2.2 Importing Pro/ENGINEER Assembly to SolidWorks 294
6.5.2.3 Importing SolidWorks Parts to Pro/ENGINEER 295
6.5.2.4 Importing SolidWorks Assembly to Pro/ENGINEER 296
6.5.2.5 Data Exchange Between CAD and CAE/CAM 296

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00006-5
Copyright © 2015 Elsevier Inc. All rights reserved.

6.5.3 Neutral File Exchange....297
6.5.3.1 IGES....297
6.5.3.2 STEP (ISO 10303)....297
6.5.4 Third-Party Translators....298
6.5.4.1 Proficiency....298
6.5.4.2 TransMagic....298
6.5.5 Solid Feature Recognition....299
6.6 Case Studies....301
6.6.1 SolidWorks Workgroup PDM....302
6.6.2 Integrated Testbed Using Windchill....303
6.6.3 Tool Integration for e-Design....305
6.6.3.1 CAD and Base Definition....306
6.6.3.2 Disciplines and Views....307
6.6.3.3 Engineering Tool Wrappers....308
6.6.3.4 Design Process Management....309
6.6.3.5 Design Collaboration....310
6.7 Summary....310
Appendix 6A: IGES File Structure and Data Format....311
Start Section....312
Global Section....312
Directory Entry Section....312
Parameter Data Section....312
Terminate Section....315
Appendix 6B: Step Data Structure and Applications Protocols....316
Questions and Exercises....318
References....319

Product data management (PDM) is the technology and associated software systems that support the management of both engineering data and process information during the product development phase and beyond. Engineering data management involves organizing, structuring, storing, and tracking the product information created by a design team while conducting engineering and product development activities. PDM aims at providing product design teams with the right data and information at the right time for making proper design decisions. There are significant benefits offered by PDM, such as interdisciplinary collaborations, reduction of product development time, reduction of the complexity in information access, and improvement of project management.

PDM is an essential subject in e-Design and a broad topic in scope. To focus the discussion, this chapter is organized with an emphasis on engineering practice and aimed at providing practicing engineers and engineering students with a brief introduction to the topic, as well as a fine understanding of the use of PDM systems to support engineering design. We narrow our scope with more emphasis on product design and less on product life cycle management (PLM). PLM is sometimes interchangeable with PDM. However, PLM is a subject of substantially larger scale. In general, PDM focuses on managing product design data as it relates to the product development phase, whereas PLM centers on reengineering product development and manufacturing processes as they relate to product life cycles. PDM is a design-focused technology that increases efficiencies within existing product

development processes by improving the management of product design data. PLM, on the other hand, is a strategic, process-centered approach that leverages PDM and other technologies to manage product life cycles, remake processes, and increase output. As a result, PLM aims at improving productivity across the enterprise rather than in a single department or a specific process (Dassault Systèmes, 2010). For a complete discussion of PLM, readers are referred to textbooks such as Stark (2011). In this chapter, our discussion stays mainly within the scope of the e-Design paradigm introduced in Chapter 1.

In addition to the discussion of PDM, we include in this chapter a highly practical and important issue in PDM—product data exchange, or more specifically, solid model translations between heterogeneous or dissimilar computer-aided design (CAD) systems (also called the interoperability issue). Solid model translations between heterogeneous CAD systems are still an ongoing research topic. We introduce the means currently available, as well as their strengths and shortfalls. Among the possible approaches, solid feature recognition (FR) has been one of the most recent developments; it provides the best possible support to address the interoperability issues within the context of e-Design. We provide a brief discussion on the underlying technology and present examples in the tutorial lessons of Projects S1 and P1 for SolidWorks and Pro/ENGINEER, respectively.

Overall, the objectives of this chapter are: (1) to provide a brief overview on PDM that introduces students to this research area and helps practicing engineers gain an understanding of PDM, (2) to present an overview of PDM software systems so that readers may explore options for software selections when an opportunity comes, (3) to discuss product data exchange and help readers understand the interoperability issue and available means for addressing it, and (4) to offer tutorial lessons that support readers in properly handling the CAD model translation issues.

6.1 INTRODUCTION

As discussed in Chapter 1, the e-Design paradigm and tool environment supports a cross-functional team for product design and development. One of the key advantages of e-Design is the intensive product data and knowledge gained in the early design stage that support better design decision-making, thus breaking the Ullman's design paradox.

In general, the amount of product data generated during the design phase is substantial. The data have the characteristics of being tentative and iterative, and intermediate with heterogeneous formats and complex relationships. Moreover, the product data often evolve along the design cycle because product development often takes significant time, especially for a complex system. The design logics and tools may vary with the development of science and technology, which leads to the revisions of data, files, and parameters. The product design team is often geographically distributed, which adds to the complexity in the management and access of product data and information. Therefore, the efficient organization and management of the massive product data becomes essential in support of product development in general, particularly when using the e-Design paradigm.

PDM is the technology and associated software systems that support the management of both engineering data and process information during the product development phase and beyond. PDM involves organizing, structuring, storing, and tracking the product information created by the product design team as they carry out engineering and development activities. With the explosion of engineering knowledge and advancement in computer-aided software tools for engineering design, PDM becomes indispensable in product development and is essential in ensuring an effective and efficient

product development process. Overall, PDM aims at providing the product design team with the right data and information at the right time, and more importantly in the right form, for carrying out engineering assignments and making proper design decisions.

In the early 1980s, many large corporations, often the original equipment manufacturers (OEMs), realized their efficiency was severely downgraded by paper-based systems. With no commercially available systems at that time, they had no choice but to develop their own data management solutions. In the late 1980s, a number of software companies started to realize the potential market of efficient data management systems and began to introduce the first generation of commercial PDM systems (Liu and Xu, 2001). The majority of those vendors at the time were already involved in the CAD/computer-aided engineering (CAE)/computer-aided manufacturing (CAM) software market, so PDM development was a natural extension of their products and service to existing customers. They focused on developing data management solutions and added PDM to their product lines (Hepplemann, 1998). Since the late 1990s, the focus has been shifted to the improvement of the product life cycles—that is, the PLM—with an aim at improving productivity across the enterprise rather than in a single department or a specific process. Also, the on-premises software from the early years has been gradually replaced with the new, alternative deployment-and-use model: the so-called cloud-based or Software as a Service (SaaS), which typically uses the Internet to remove the need for the user to install any software on premises. Such software offers benefits, such as running software remotely, which can result in considerable cost savings because of reduced staffing, maintenance, and other factors.

PDM systems are increasingly being used in industrial applications for long-term archival of product information, as well as to enhance collaboration and communication throughout the design process, support distributed design teams through advanced document sharing, track changes in product information, and control design documents (ranging from requirements information to CAD). The adoption of PDM systems has caused a change in how design processes are managed and how individual designers collaborate. Additionally, companies are pushing the limits of currently available PDM software, resulting in continual development and new application domains. For example, commercial software vendors have integrated PDM systems with other design support tools, included automated workflow management and suites of CAD/CAM/CAE tools, and refined and developed new functionality (Caldwell and Mocko, 2008).

Today, PLM is primarily used in the automotive and aerospace industries, as well as in the machinery industry (Abramovici and Sieg, 2002; Lee et al., 2008). For example, GM credits PLM initiatives with decreasing time-to-market from 48 to 18 months (Tang and Qian, 2008). Automotive industry leaders such as Autoliv, Eaton, Honda, and Johnson Controls are driving success by using the MatrixONE solutions (Tang and Qian, 2008). Regarding the importance of PLM to the automotive industry, Reale and Burkett concluded that ‘‘the smarter the car, the more automakers need PLM’’ (Tang and Qian, 2008, p. 288). Among many successful stories, we include two of the most notable in Section 6.3.4 to illustrate a few insights.

Although PLM is meant to manage product information throughout the entire life cycle of a product, an international study revealed that the adoption of PLM is still mainly limited to product design (Abramovici and Sieg, 2002). Today, PLM is an active research topic, especially in supply chain integration and the integration of business process into the overall product life cycle development. Nevertheless, in practice, PDM and PLM are often interchangeable, particularly from a product development perspective.

In practice, product data are largely embedded in files. Many student teams may not have access to full-range PDM or PLM systems and must reply on ad-hoc approaches for file management. We include a number of commonly employed approaches to support file management in Section 6.2.1. We also include a case study in Section 6.6.1 to illustrate the use of SolidWorks Workgroup PDM, a mid-range PDM system, which is commonly available to students.

If you work with multiple CAD systems, you might need to translate solid models from one CAD system to another for numerous purposes. You may need to bring parts from CAD system A to CAD system B in order to conduct Finite Element Analysis (FEA), generate a machining toolpath, or perform other engineering activities. You may be given an assignment to bring in parts and sub-assemblies from other CAD systems to the major CAD software your company is using in order to generate a complete product model in the designated CAD system. In industry, the OEM integrates and communicates with their suppliers, during which they may encounter the issue that the CAD models created by suppliers may not be compatible with the major CAD system used by the OEM. CAD model translation (also called interoperability among CAD systems) is a practical and essential issue in product data integration for engineering design. The question is how to handle CAD model translations. What are the available options? How practical are these options? Therefore, in addition to discussing the PDM, we address this practical and important interoperability issue encountered in product data exchange—that is, CAD model translations between heterogeneous (or dissimilar) systems that employ different geometric kernels.

In this chapter, we address two issues that are very relevant to product design. First, we provide an overview on the practical means for file management and an introduction to PDM technology and systems. We start by discussing file management in Section 6.2. In Section 6.3, we present the fundamentals of PDM, including the product and process data that PDM manages, the functionalities of PDM, and the benefits and successful stories of PDM technology. In Section 6.4, we discuss commercially available PDM systems. Then, in Section 6.5, we shift focus to the second issue—the product data exchange. We discuss the viable options and available model translators, as well as their strengths and limitations. We include examples that outline the part and assembly model translations between Pro/ENGINEER and SolidWorks, as well as feature recognition (FR) in both Pro/ENGINEER and SolidWorks. In Section 6.6, we offer case studies that illustrate the practical use of PDM, including SolidWorks Workgroup PDM.

6.2 FILE MANAGEMENT

During the product development phase, a large amount of data is generated. This includes product data and process data in the forms of files, documents, and diagrams, etc. Typical product-related data include CAD geometry, engineering drawing, specifications, project plans, part and assembly files, bill of materials (BOM), engineering simulation data, engineering change requests, and so forth, which are shared throughout the product development phase by the product development team and by the extended enterprise. The product-related data and information are stored in the forms of paper documents, digital files, and information extracted and stored in databases. In the context of e-Design, digital files are the most common and largest in quantity, including document files such as specifications, configuration, and purchase orders; product models, such as CAD drawings, part files, and assembly files; CAE analysis model and result files; and manufacturing-related information, such as numerical control (NC) programs.

The management of information in modern engineering design projects is typically characterized by the following (Caldwell and Mocko, 2008):

- A large amount of digital product information, generated by different engineering software, is often stored in a variety of formats.
- Documents often go through several revisions during the product development phase, which may be initiated and completed by different designers and engineers of different disciplines. Access to documents must be controlled across members of the design team.
- Documents are highly interrelated, such that the changes in a single document may be propagated through several other documents.
- Data sharing and design collaboration take place between design team members in distributed locations.

Nevertheless, the most basic function of a PDM system is digital file management and sharing. In any team design projects, large or small in scale, file management and sharing among the team members is critical; it should be the first issue addressed at the onset of the project. PDM certainly offers excellent capabilities in support of file management. However, in academic environments, student teams may not have access to full-range PDM systems to support their design activities. Therefore, in this section, we first discuss file management without a PDM and then provide a short introduction to the file management aspect of PDM systems. In Section 6.6.1, we discuss SolidWorks Workgroup PDM, which is part of SolidWorks Premium and Professional package and is popular among engineering students.

6.2.1 AD-HOC METHODS

Product data can be shared by using a variety of technologies, including email, web-based workspaces (i.e. Google Groups, Yahoo Groups), shared network drives with file management systems, and PDM systems, to name a few. In this subsection, common ad-hoc methods are discussed for sharing product data embedded in digital files. These methods may be useful to student teams for exchanging product data in collaborative design projects.

Ad-hoc methods of file management work reasonably well for self-contained files, such as Word documents and Excel spreadsheets. However, they break down quickly for more complex CAD file management. Some of the most common methods (Buchal, 2006) include email file attachments, peer-to-peer file sharing (e.g., Windows Messenger), removable media (CD-R or USB drive), FTP, shared network folders, web folders, and Microsoft SharePoint.

Email file attachments, peer-to-peer file sharing, and removable media are simple and widely employed. However, this type of management is difficult to maintain and does not fully support multiple users accessing data from distributed systems. The main issue is that distributing a file to multiple recipients immediately creates multiple instances of a file, with no mechanism for version control or reconciliation. In some cases, recipients may encounter problems in reviewing files. For example, if a CAD assembly file is sent, it may not be opened because the referenced component files may not be located properly due to incorrect drive letter and/or folder path.

Files may be shared using standard network communication and shared file folders, such as shared network folders, web folders, and Microsoft SharePoint. These approaches provide a simple way to share files within a team, either via a network or the web browser, while offering access control and

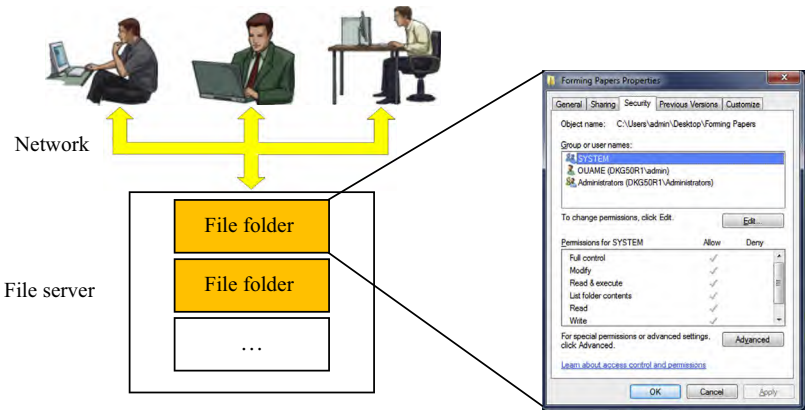


FIGURE 6.1

Controlled access to shared network drives.

basic check-in/check-out. For example, using network folders, if a user has a file open for editing, it becomes available as read-only to other team members. Additionally, it is possible to control file and folder using security properties in Windows operating systems (see Figure 6.1). Access is restricted to computers connected to a Windows LAN domain, so students with laptops or home computers may not access the shared folder. On the other hand, anyone with an Internet connection and user account can access web folders. Web folders provide access control and allow editing in place without downloading and uploading. Some software supports accessing to files via network, such as the Open from Web Folder dialog box in SolidWorks. The assembly files and related part files are all located on the web server, and they are accessed using a URL rather than a drive letter and network path. However, SolidWorks web folders maintain product structure, but offer no version management or check-in/check-out capabilities.

SharePoint is a web-based shared workspace with many collaboration tools, which provides access control, check-in/check-out, revision management, and many other collaboration capabilities. SharePoint is not designed to manage product structure, so files can only be opened or downloaded individually (Caldwell and Mocko, 2008). Figure 6.2 shows a view of a SharePoint document library containing SolidWorks part and assembly files (Buchal, 2006). It is worth noting that SharePoint document libraries can be accessed as web folders from SolidWorks, but SharePoint version management and check-in/check-out are not integrated with SolidWorks.

Another important issue encountered by a project team without access to a PDM system is viewing the product model. On many occasions, team members need to view CAD models without the ability to change them or to incorporate them into other models. Some viewers, developed by CAD vendors, can open and view CAD files in their respective native formats and more; for example, the NX viewer views NX part and assembly, I-DEAS files, Parasolid, and JT files, whereas the SolidWorks viewer views SolidWorks part and assembly and JT files. Viewers allow designers to zoom, pan, and rotate models. Some viewers allow users to take sections with multiple section planes. Some support users to add notes, text with leaders, and dimensions. Most viewers incorporate markups and save them as JT format.

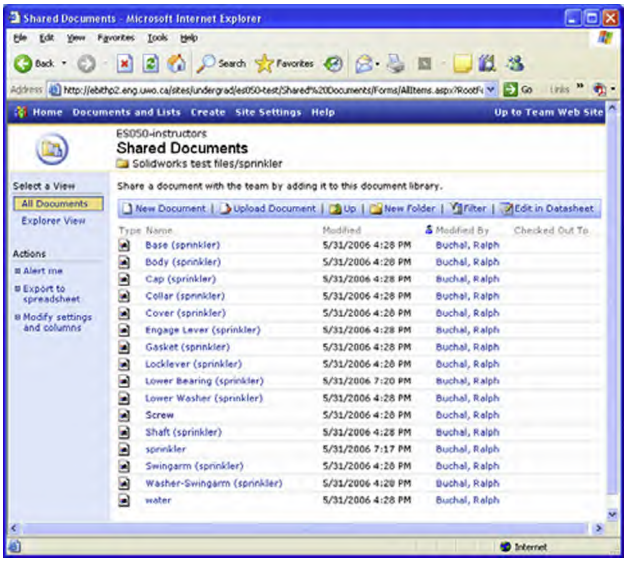


FIGURE 6.2

Document library view in SharePoint (Buchal, 2006).

Many viewers support JT files, which are a three-dimensional (3D) data format developed by Siemens PLM Software (formerly UGS) and used for product visualization, collaboration, and CAD data exchange to some extent. JT viewer (Figure 6.3) is free for download and was recently extended to support iPad, iPhone, and iPod Touch (JT2Go), which moves mobile engineering design one step forward.

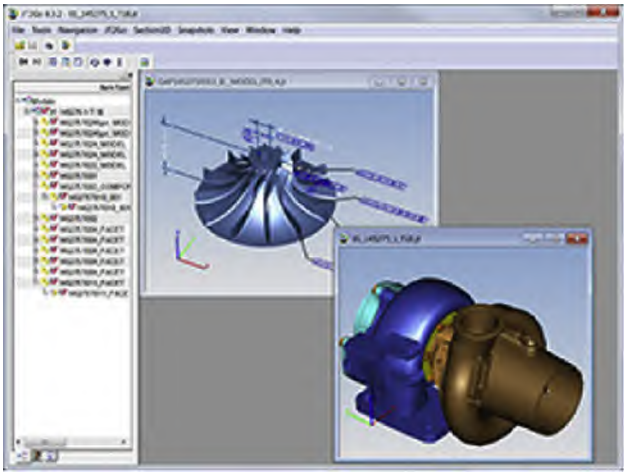


FIGURE 6.3

Computer-aided design viewers. JT viewer (www.plm.automation.siemens.com/en_us/products/teamcenter/lifecycle-visualization/jt2go).

Although ad-hoc approaches to file management support design teams in managing files and sharing design data to some extent, they are less desirable in practice. There is not a standardized revision system. Revisions made to documents are not structured. Changes between documents are not explicitly captured, thus losing the evolution and refinement between documents. There is not a means for controlling specific aspects of a file and/or contents within the file. In addition, standardized locations that are accessible by all designers may not exist.

In the next section, PDM systems are presented as a means for addressing the problems associated with information management using a traditional file-based approach. Please note that, in general, PDM has a lot more capabilities than just managing files.

6.2.2 PDM APPROACH

PDM systems attempt to address file management issues by (1) structuring existing document meta-information, (2) adding meta-information, and (3) enforcing rules for creating, accessing, sharing, and modifying documents. Specifically, PDM systems provide greater control for enforcing effective document management practices through the use of revisions, locations, editors, owners, and much more information about the data stored within a file. Metadata, in a digital context, are the data used for describing the file or the content of a document. The meta-database in a PDM system controls the document's relation to other documents and the rules for how the system links information. The basics of how PDM systems work in managing files are illustrated in Figure 6.4.

PDM systems offer functionality at the server side and the client side. Typical PDM systems provide controlled access to document vaults using secured login. Individual designers have controlled access ability to check-in/check-out documents to ensure changes are not being concurrently made

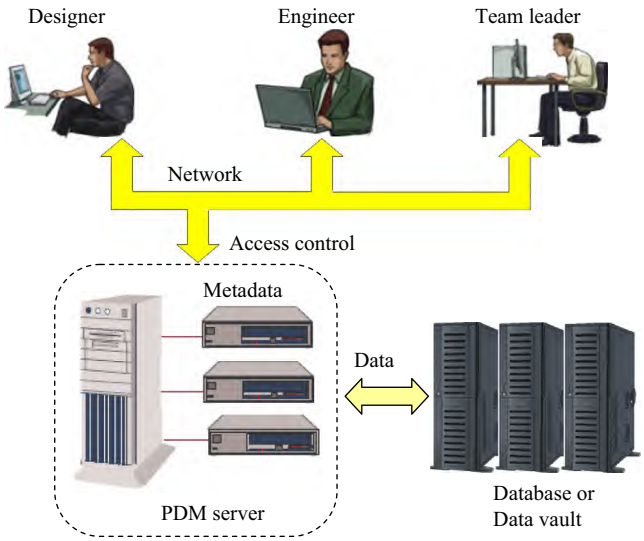


FIGURE 6.4

Architecture of PDM systems.

resulting in document conflicts. They also support workflow management and trigger for automated document tracking and notification to design team members. PDM systems provide a means for flagging changes for affected team members for document review, approval, and release. They provide storage of documents in a shared location with the ability to create specialized locations for projects and groups, and they support the query and retrieval of documents based on richer document descriptions. They also standardize revision schemes, which is essential for data management. In addition to enabling document sharing, PDM systems are typically integrated closely with CAD software as integrated add-ons or stand-alone applications. More about commercial PDM systems is discussed in Section 6.4.

6.3 FUNDAMENTALS OF PDM

PDM forms the product information backbone for a company and its extended enterprise, which allows the cross-functional team to contribute throughout the product design and development phase. In addition to file management, PDM can be viewed as a data or information integration tool connecting different functional areas. It ensures that the right data and information are available to the right person at the right time—and more importantly, in the right form. In addition, PDM improves communication and collaboration between groups of diverse functions and engineering expertise in the enterprise. The area of application of a typical PDM system is shown in Figure 6.5 (CIMdata, 1998).

Although it is highly desirable that knowledge is accessible when a design decision is to be made, in reality, knowledge is not directly available but is obtained by interpretation of information deduced from analysis of data. In general, data are available to an organization in the form of observations, computational results, and factual quantities. Interpretation, abstraction, or association of these data leads to generation of information. Finally, knowledge is obtained by experiencing and learning from this information and putting it into action (Owen and Horváth, 2002). In fact, looking at engineering design from a teleological point of view, it can be said that the primary function of engineering design

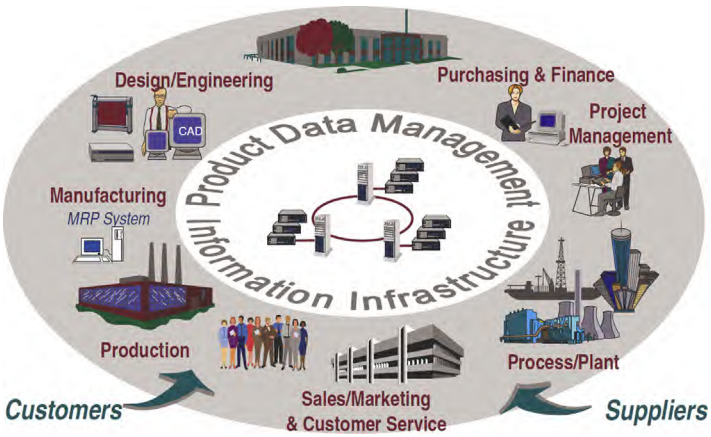


FIGURE 6.5

Application of a PDM system in an organization (CIMdata, 1998).

research should be to transform empirical or rational knowledge into a form that can be used for practical deployment (Horváth, 2004). Design information extraction and knowledge management is an ongoing research topic. Interested readers are referred to excellent review articles as an introduction, such as Chandrasegaran et al. (2013).

Our goal in this section is to provide readers with a fundamental understanding of PDM from the context of e-Design. In Section 6.3.1, we discuss the product data model that describes the data and its structure to be managed in support of e-Design. In Section 6.3.2, we discuss the basic functions of a PDM system that supports the design team in managing the product data. In Sections 6.3.3 and 6.3.4, we present the benefits that PDM offers and the impact of PDM to industry, respectively.

6.3.1 ENGINEERING DATA MODELS

Engineering data models, which consist of product data models and process data models, provide the engineering team with a consistent and unified engineering data set that supports engineering activities for product development. A product data model is evolving throughout the product development process and beyond; for example, a revolution of CAD models of a high-mobility multipurpose wheeled-vehicle (HMMWV) throughout the design phase is shown in Figure 6.6. On the other hand, a process data model is relatively less involved.

6.3.1.1 Product Data Model

In general, the size and contents of the product data can be different from one product to another, depending on many factors, such as the nature of the product being developed, the design process employed, and the tools and technology adopted for the product development, among others.

From the data authoring perspective, product data can be categorized into three types: documents, files, and data or parameters. Documents are usually authored by engineers, such as product requirement and specifications, organizational structure of the product development team, major milestone and workflow, reports, guidelines, standards, and manuals. Documents can be in the form of Microsoft Word, PowerPoint, Excel, or Project files, or as Adobe PDFs. In addition, pictures and videos in numerous formats are created to support visual aid of the product in the design phase. In e-Design, a large amount of files are created by software tools. Geometric model files, including part and assembly files, are created by CAD and exported in other formats, such as IGES (IGES, 2001) or

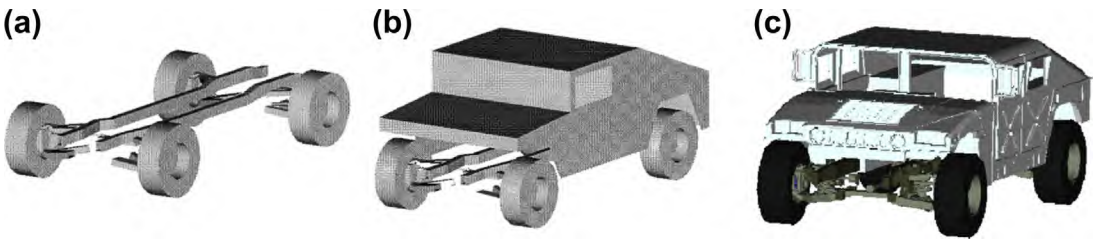


FIGURE 6.6

HMMWV CAD model for concept and detailed designs. (a) A 15-part concept model, (b) an 18-part model for intermediate design, and (c) a detailed design model with more than 200 parts (Chang et al., 1998).

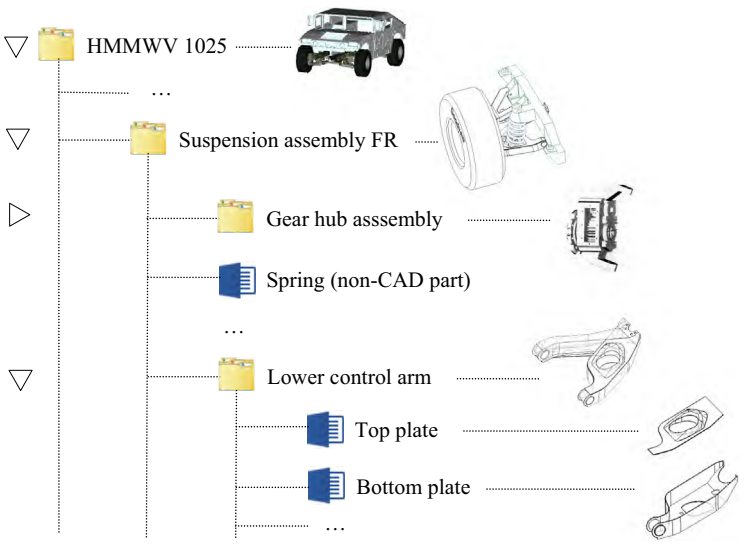


FIGURE 6.7

Example of bill of materials for an HMMWV.

STL (en.wikipedia.org/wiki/STL_(file_format)), for numerous purposes. Simulation model files are created for engineering analysis, such as FEA, and the result files are generated by the respective FEA or simulation software tools. In addition, toolpath files and machining model are generated using CAM software. Data or parameters extracted from files or documents or entered by the product development team are stored in the database, such as the BOM of the product extracted from the CAD model of the product.

All these documents, files, and data must be structured and organized to support the needs of product design, including product data model and process data model. There are many different ways to structure and organize the data, as long as the data models are logical and facilitate the product development team in accessing product information in a timely manner. One of the most common ways to structure and organize product data is using the BOMs. For example, a BOM of the HMMWV is shown in Figure 6.7, in which the product is broken down into parts and subassemblies. Each entity is given a name (and identification number) and icon that links to more data and information.

Data can be linked by using, for example, HyperText Markup Language (HTML) for creating web pages and other information that can be displayed in a web browser. For example, the BOM shown in Figure 6.7 is displayed in a scrollable window (shown in Figure 6.8(a) with each entity name implemented as a hypertext). Once clicked, its associated data are displayed in the window below. For example, if Suspension Assembly FR is clicked, its geometry is brought into a viewer (left window) and its properties (e.g., mass properties) appear in the right window. Engineers may pan, rotate, and zoom in/out the model in the viewer to gain a better understanding of the model geometry and its constituent components. In addition, right-clicking an entity brings up its associated page for more information. For example, right clicking the HMMWV 1025 at the top of the BOM in Figure 6.8(a) brings up the product development page shown in Figure 6.8(b). In this page, the introductory

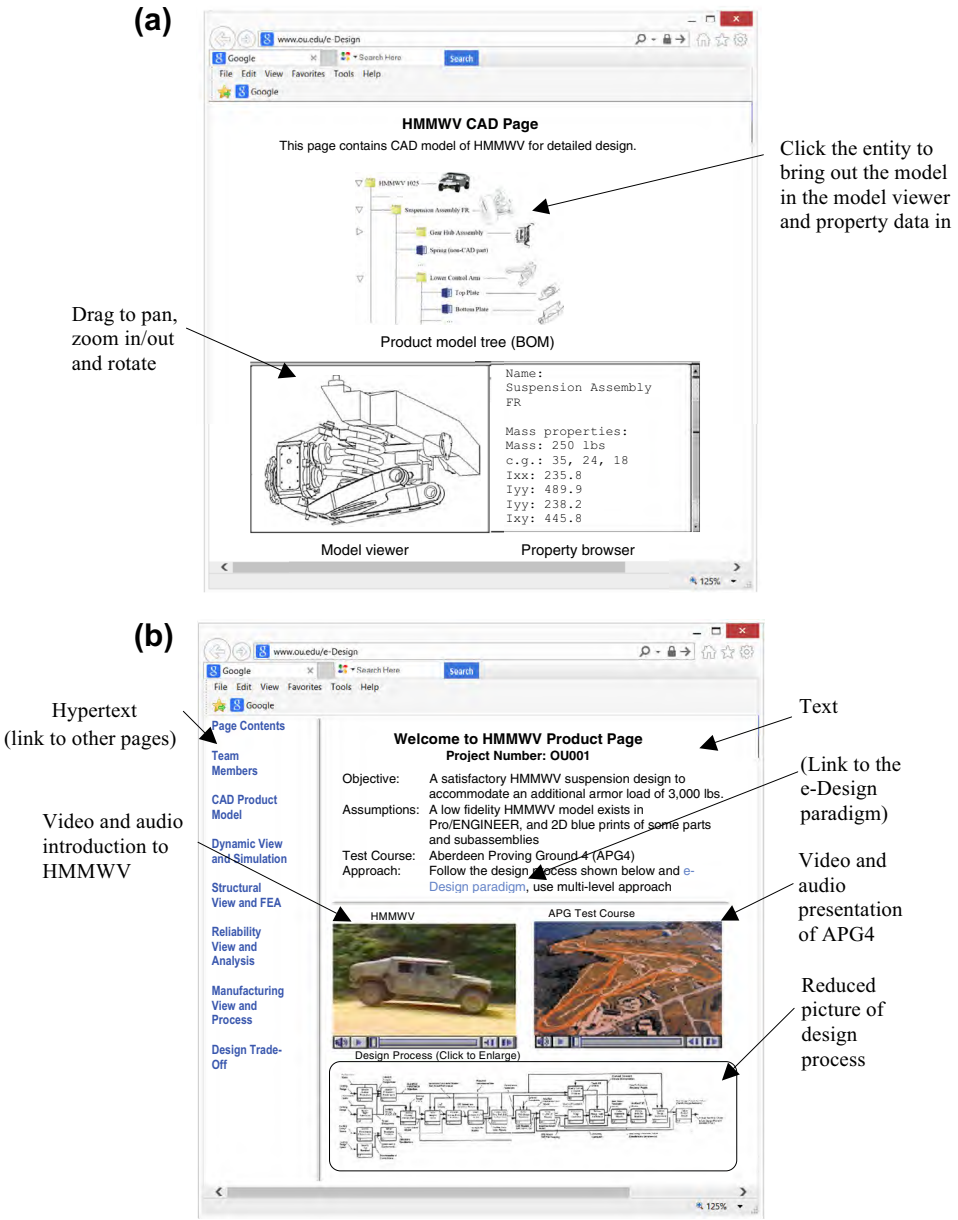


FIGURE 6.8

Sample web-browser pages of PDM of a product data model. (a) HMMWV CAD model page, and (b) HMMWV product page.

information about the design project, such as objective and assumptions, are listed, together with videos offering visual aid to the HMMWV physical model and the test course. In addition, an overall design process is defined and shown in the bottom half of the page. Hypertext on the left connects to other pages, such as connecting back to the CAD model page shown in Figure 6.8(a).

A typical PDM system supports such product and process data models in some way. Even without a commercial PDM system, implementing such web pages for support of PDM is not too difficult. For those who took courses such as engineering multimedia, creating data management pages like those of Figure 6.8 is generally straightforward. The model viewer on the lower left portion of the CAD model page (Figure 6.8(a)) can be implemented using JT Viewer (http://www.plm.automation.siemens.com/en_us/products/teamcenter/lifecycle-visualization/jt2go) or Virtual Reality Modeling Language (VRML; en.wikipedia.org/wiki/VRML) viewers, for example. You may publish your solid models (part or assembly) as JT or VRML models. These files can be stored on a website or emailed as an attachment. Anyone can view the JT or VRML files by using a free plug-in to their browser, such as Cosmo VRML Player for Windows; view3dscene for Windows, Macintosh, and Linux; or JT viewer for JT models.

The BOM shown in Figure 6.7 is the most basic product data model, which provides information that helps team members gain a first-level understanding of the product being developed at a given time. In e-Design, the product data model should support follow-up activities, including engineering modeling and analysis, manufacturing process and machining simulations, and design trade-offs. A BOM is just a start, and a BOM alone is not sufficient to support all design activities.

How do we proceed with the e-Design paradigm from here? There are at least three options: (1) using commercial CAD/CAE/CAM, (2) using commercial PDM, and (3) developing one's own tool and information integration infrastructure. The option of using a commercial CAD/CAE/CAM suite is for those who have access and are able to depend on the engineering capabilities offered in the software for support of engineering design. Software suites, such as SolidWorks (with SolidWorks Motion, SolidWorks Simulation, and CAMWorks), Pro/ENGINEER (with Pro/MECHANICA Structure, Pro/Mechanism, and Pro/MFG), or CATIA (with FEA and CAM modules) may be used for this option. In this case, the design team may proceed from CAD to CAE and CAM in a straightforward fashion because the transition from CAD solid models to CAE simulation or CAM toolpath generation is seamless. There is no need to do anything extra when going forward to carry out CAE and CAM activities. Most PDM is taken care of by the commercial software. This option is ideal for a small project team that is dealing with relatively smaller-scale design projects, such as capstone design projects for senior engineering students. The design team will have to manually organize the product data beyond the analysis phase, such as conducting design changes across disciplines. One key condition for using the commercial CAD/CAE/CAM software is that the CAE and CAM capabilities offered must be able to support all engineering analysis requirements for the design problem at hand. If this is not the case (e.g., the design problem involves structural analysis of engine mounts made of rubber and none of the CAD/CAE/CAM suites offer FEA for rubber), then an FEA code that is capable of supporting the required analysis, such as ABAQUS (www.3ds.com/products-services/simulia/portfolio/abaqus/overview), must be employed. In that case, the CAD model of the engine mount must be imported into ABAQUS (or other modeling tools, such as PATRAN or Hypermesh) for mesh generation, and loads obtained from motion analysis must be converted into a format that can be incorporated to the FEA model in ABAQUS for analysis. This approach is suitable for a design problem that heavily involves engineering analysis.

The second option is using commercial PDM (or PLM) software, such as Windchill from Parametric Technologies Corporation (PTC) (www.ptc.com), to support product data integration without a fully integrated CAD/CAE/CAM software suite. In this case, the design team will have to manually handle simulation model generation based on the product data embedded in the CAD solid models. For example, in order to create a motion simulation model using, for instance, ADAMS (www.mscsoftware.com/product/adams) or DADS (www.lmsintl.com), the design team will have to first organize the product CAD model into subassemblies or parts as individual bodies and collect mass properties of individual bodies with their respective coordinate systems. Then, the team will use the mass properties provided by CAD models to create a motion simulation model using ADAMS or DADS externally to the CAD software. Some software, such as ADAMS/Car (www.mscsoftware.com/product/adamscar), offers templates that facilitate the creation of a motion model. This approach is suitable for a design problem that is less involved with engineering analysis and more dependent on the design process and data integration. An example of such an application is provided in Section 6.6.2 as a case study, in which an integrated testbed for reverse engineering using Windchill is discussed.

The third option is developing a tool and information integration infrastructure to support product development, in which a customized software infrastructure is implemented to integrate CAD, CAE, and CAM software and provide design trade-off and product management capabilities to meet specific product development needs. This is certainly not a trivial task. However, this approach offers maximum flexibility for a cross-functional team in product development. An example of such an application is provided in Section 6.6.3, in which a software infrastructure of tool integration for e-Design developed a few years ago is presented.

6.3.1.2 Process Data Model

In general, process management in a PDM system supports the engineering team in defining, disseminating, coordinating, and tracking design activities. The design processes are often described and modeled in a flow chart. The modeled workflow is then executed and the PDM system manages the actual workflow, so that the right work is done at the right time with the right information by the right person (Lee et al., 2010). The PDM system gives notifications to control the activities. For example, a designed part that has not been approved yet by the management will not be manufactured.

There are two types of workflow: static and dynamic. Static workflows are fixed; once they are modeled and started, they have to be finished according to the model. Dynamic models can be modified easily because usually there is a visual flow chart that can be used in a drag-and-drop style. Once a dynamic model has been started, it can be changed if the process needs to be changed while the workflow is in progress (Qiu and Wong, 2007). Older PDM systems usually apply the schematic static workflow, whereas the newer systems use graphical workflow modeling. After modeling the workflow, permissions are assigned to different users, allowing them to approve, release, or modify the documents.

As discussed earlier, a PDM system also serves as a process data management tool. A logical way to organize process data is associating the data with the design process. In general, design process can be defined in different levels. One task, represented in one block in the process chart of a higher level, can be expanded into more detailed tasks in the lower-level process, as illustrated in the upper box of the web page shown in Figure 6.9. When an entity of a task is clicked, the task-associated data are

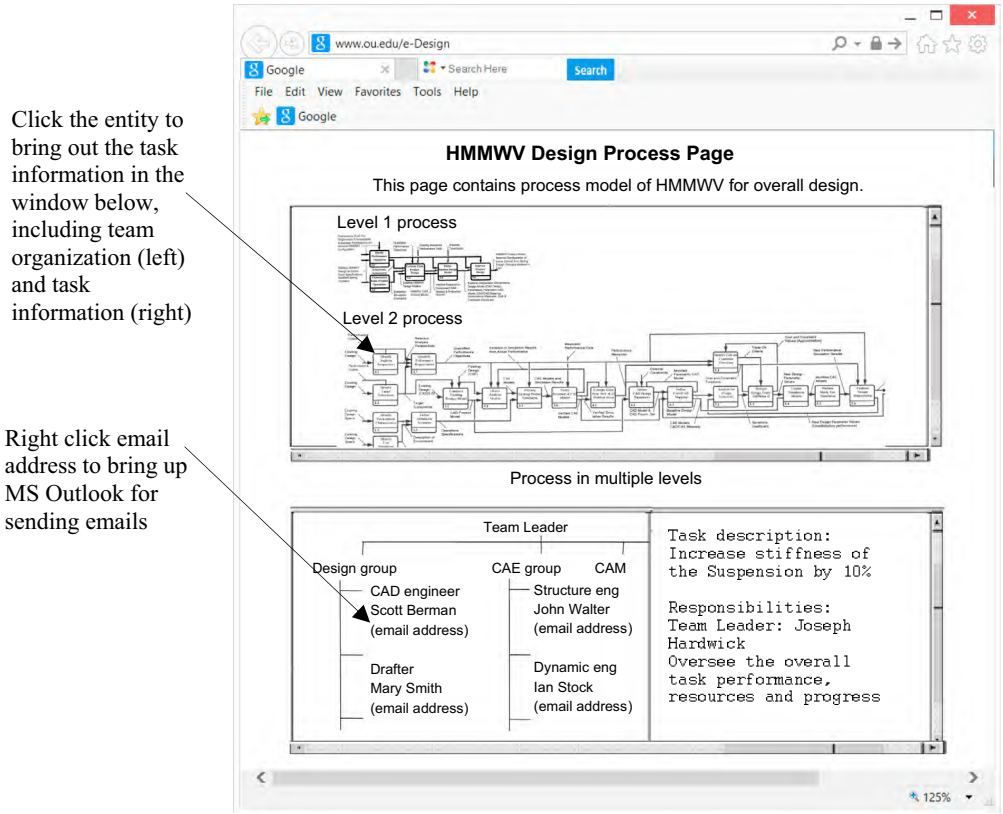


FIGURE 6.9

Sample web-browser page of PDM for process data model.

displayed in the window below. For example, if a top-level task is clicked, the organizational structure of the project team that is in charge of the task is displayed in a viewer (left window) and the task assignments, personnel, and individual assignment and responsibility are displayed in the right window. Right-clicking an email address below the name of the team member brings a Microsoft Outlook page for sending email to the person.

6.3.2 BASIC FUNCTIONS OF PDM SYSTEMS

Although there is no clear consensus in industry and academia regarding the functionality of PDM systems, many articles refer to CIMdata (1998) and Crnkovic et al. (2003) for the common denominator. According to Crnkovic et al. (2003), the functionality of PDM systems can be grouped into two categories: user functions and utility functions. The user functions allow users to interact with the PDM system either as a user or as an author of information. The utility functions connect to the network infrastructure and support user functions by providing interfaces between different operating environments.

6.3.2.1 User Functions

These user functions include data vault and document management, workflow and process management, as well as program management.

6.3.2.1.1 Data Vault and Document Management

The core element in a PDM system, which is closely related to document management, is vault. Vault, from a logical point of view, is a single place where all documents are stored. Typically, it is a computer server (or group of servers) that physically stores the documents that are encompassed by operation of PDM. The documents in vault are accessible only by PDM users through PDM client functions. After the document is created on a computer, the operation "check-in" is performed, which transfers the document from the client computer to the vault. From this point on, the management of the document is seized by PDM and all further actions involving this document have to be performed by using the PDM client functions. If a team member wants to access this document, the operation "check-out" should be performed. Documents stored in the vault can be reviewed without checking out. An example of document processing using a PDM system is illustrated in Figure 6.10.

Because a PDM system usually involves many users, there will be situations when two or more persons want the same information at the same time. A PDM system controls the access to information and to what extent the information is available. Therefore, another important aspect of document management in PDM is version and status. In a typical PDM system, when a document is sent to the vault by check-in operation, it receives the status of "checked-in," meaning that the document is ready for next actions. The first person to access the current information, such as a Word document, checks it out and becomes the temporary owner of the information. If another person tries to access the same information, the information will either be blocked or made available as a read-only copy. This state is maintained until the first person checks the original information back in again. After a cycle of processing, the document status can be changed to "Released," indicating that the document obtained a satisfactory status and is ready for general use. The history of changes, including dates, persons, etc., together with all versions that were created during such a cycle, are collected by the PDM system. An example of the cycle is illustrated in Figure 6.11.

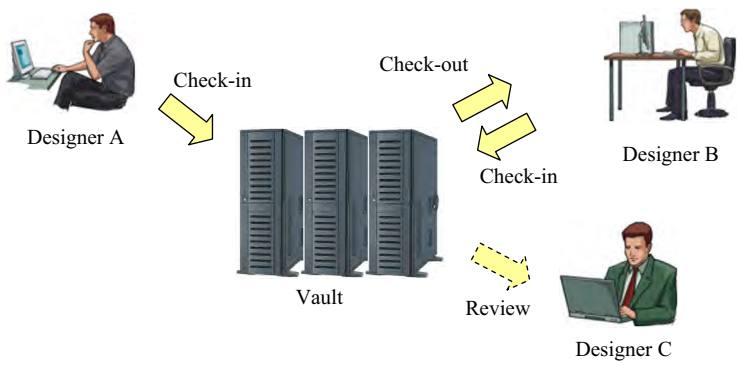


FIGURE 6.10

Example of document processing in a PDM system.

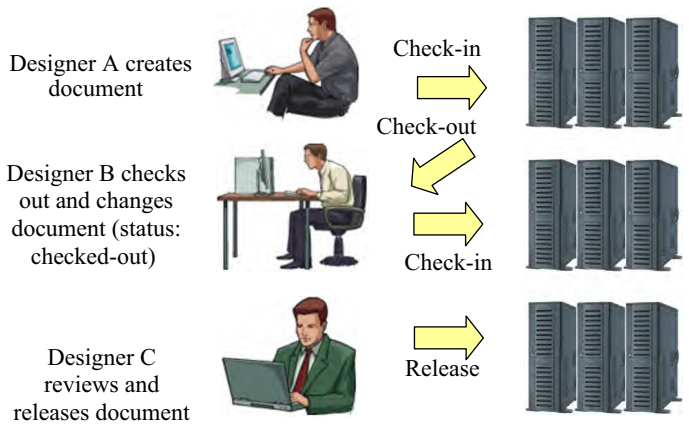


FIGURE 6.11

Document status and versions during processing in a PDM system.

6.3.2.1.2 Workflow and Process Management

Workflow and process management is used to define and control the workflows and information flows, such as engineering change procedures and release procedures. In PDM workflow management, these processes can be modeled and managed, allowing automated distribution of the right information to the right users. Discussion of Figure 6.9 presents a practical implementation for work and process management.

6.3.2.1.3 Program and Project Management

Program management connects the product data with project data, thus allowing resource allocation and project tracking. In this way, the projects using a specific product or part can be found. Program and project management issues might involve planning and work performance control. Project leaders can overview the effort and performance of a project team and check if the progress is within the time schedule.

6.3.2.2 Utility Functions

Underlying those user functions are the utility functions. They are connected to the network infrastructure, and insulate the end user from that. The utility functions provide interfaces between different operating environments and include communication and notification, data transport and translation, image services, administration, and application integration.

6.3.2.2.1 Communication and Notification

Within an organization, different users are interested in different information. PDM systems can provide notification features that inform certain users when a specific event occurs (e.g., a task is finished or the project state has changed), in addition to initiating and organizing web meetings. The users who are to be informed about the event can be connected to roles and assignments of the individual team members. The notification is commonly sent as an email.

6.3.2.2.2 Data Transport and Translation

Sometimes, data need to be transported from one subgroup to another. On many occasions, data need to be translated from one engineering tool to another. These tools may not support the same data formats. By providing a translation service, a PDM system can overcome this problem. The translation can be performed manually or automatically. More about product data exchange and CAD model translations is discussed in Section 6.5.

6.3.2.2.3 Image Services

In the product development process, CAD and different engineering tools are used to model a product. For easier access to the product model and information, the PDM system provides tools or add-ons that, for instance, allow models stored in a CAD system to be viewed from the PDM system. Model viewers discussed in Section 6.2 represent a typical scenario of the image service function in a PDM system.

6.3.2.2.4 Administration

Because PDM systems are quite extensive, they require a lot of administration. In addition to the usual administration tasks such as installation, configuration, maintenance, user authorization, role management, and data backup, the PDM administrator defines workflows, translations, and tailors the system for the company.

6.3.2.2.5 Tool Integration

In PDM, it is important to collect all data in one location in order to avoid data inconsistency. The integration between engineering tools and the PDM system is important to enable this. In Section 6.3.1, three options were discussed to address the integration between CAD/CAE/CAM and the PDM system. In Section 6.6.3, we present a software infrastructure that supports tool integration for e-Design.

6.3.3 BENEFITS OF PDM SYSTEMS

In general, organizations that successfully implement PDM can achieve multiple advantages in terms of productivity and competitiveness. The benefits can be summarized as follows (CIMdata, 1998; Miller, 1998).

- *Collaboration between design team members.* PDM software provides a virtual workspace in which design team members can store and share documents related to projects. Additionally, interdisciplinary collaboration between designers, marketing, and manufacturing is supported. A PDM system can lead to collaborative development of new products, as well as improvements on existing products.
- *Reduced product development cycle time.* Due to the increased collaboration with all areas of an organization and its supply chain, as well as the easy access to product information, the product development time can be greatly reduced. This enables organizations to respond to the market with greater effectiveness and consistently provide their customers with new and innovative products. In addition, when properly implemented, PDM can simplify many day-to-day user operations by managing and automating routine tasks, such as searching for drawings, tracking approvals, and completing status reports. This improvement dramatically decreases the user's non-value-added time.

- *Workflow and project management.* Project management is made easier using a PDM system because all those involved in the project have access to the same information and can work with a common product model. A PDM system also allows project managers to track the progress of a project more effectively and therefore ensure that the work being carried out is correct, on schedule, and on target.
- *Improved life cycle design.* The information captured within PDM systems is increasing from solely a CAD focus to include several engineering domains, and it supports easy access to information on new product development. It allows manufacturing staff and production engineers to access design information at a much earlier stage of the product development, hence making it easier for problems to be identified earlier rather than later.
- *Supply chain collaboration.* PDM systems are considered to have a strong impact on supply chain relationships by linking subcontractors, vendors, consultants, partners, and customers and giving them access to the same information. PDM systems can also act as a data store for internally developed parts and external parts available from suppliers. By using a PDM system's database of existing parts, a designer can eliminate duplicate work and therefore considerably reduce development time and cost.

6.3.4 IMPACT TO INDUSTRY

As mentioned earlier, the automotive and aerospace industries are the biggest adopters of PLM. The high degree of penetration of PLM in the automotive and aerospace industries is due to the fact that their products have long life cycles, are very complex, and have nearly no possibility of physical prototyping (Liu and Xu, 2001). We briefly mention two notable success stories in using PDM or PLM for product or project development: Boeing 777 and Ford C3P; both are extracted mostly from Caldwell and Mocko (2008).

As reported by Caldwell and Mocko (2008), Boeing began using 3D solid modeling software and a PDM system during the design of the 777 jet aircraft. Previously, Boeing used two-dimensional (2D) modeling software to design its airplanes, which required many stages of design of parts and subsystems of the airplane in order to ensure that all components fit together properly. Boeing used three stages of mock-up before producing a final design; even with three rounds of mock-ups, the final design would have parts with mismatched geometries. For the design of the 777 jet aircraft, Boeing implemented a new 3D CAD system, which consisted of CATIA and Electronic Preassembly Integration (EPIC) on CATIA. These two programs helped Boeing eliminate mock-ups by allowing parts to be designed and assembled together in the computer. This allowed Boeing to ensure that part geometries would match up properly. By converting to a 3D CAD system with assemblies, Boeing was able to speed up its design process and eliminate many errors. Boeing also took advantage of the all-computer design by using a PDM system. Boeing stored all of its CATIA files on the world's largest (at the time) grouping of IBM mainframe computers in Bellevue, Washington. This allowed companies in Japan, the United States, and the United Kingdom who were working with Boeing to access the CAD files at any time. These suppliers, therefore, were aware of changes made to the design very soon after the changes were made. The implementation of this new system allowed Boeing to reduce engineering change requests by 90%, reduce cycle time for these requests by 50%, reduce material rework by 90%, and improve fuselage tolerances by 5000%.

Another well-known story is the C3P initiative at Ford Motor Co. As reported by Caldwell and Mocko (2008), Ford Motor Company implemented a PDM system worldwide in the 1990s. It was part of a new CAD initiative that Ford called C3P, an acronym for CAD/CAM/CAE/Product Information Management (PIM). PIM is a Ford-specific term for PDM. Although this project was called C3P, Ford's focus in this endeavor was on the PDM system. This project began in 1996; by mid-1998, 16 vehicle programs had already begun using the new system. Ford's plans were to use the PDM system worldwide, so that all of its operations and suppliers accessed CAD files that were stored in Dearborn, Michigan. Ford's C3P program was a $200 million deal with SDRC, which included both software and services. Before this deal, Ford had used other CAD and PDM systems, including a PDM system developed in-house. Ford experienced the obvious improvement, which was a faster time-to-market of its products. Engineering efficiency rose around 30–40% due to new solid modeling capabilities. Prototype costs decreased by 40–50%, saving hundreds of millions of dollars. Late changes were reduced by 50%, and programs were able to be completed in less than 2 years. Ford was able to extend the benefits of C3P beyond the design of the vehicle itself. They used computer programs to analyze the solid models in order to determine a vehicle's manufacturability within an existing plant. In one case, Ford was able to prevent a $60 million tooling modification that would have been required had the design not been analyzed ahead of time. Ford's C3P program was a success that continues today. Thus, by implementing a PDM system, Ford was able to reduce time-to-market of new vehicles, increase engineering efficiency, reduce prototype costs, and reduce late changes to parts.

6.4 PDM SYSTEMS

From the mid-1980s through the late-1990s, we saw the development of many capable PDM systems, including iMAN from UG Solutions, Metaphase from SDRC, Optegra and Windchill from PTC, MatrixOne, Pro/PDM and Pro/Intralink from PTC, ENOVIA from Dassault Systèmes, and Workgroup PDM and Enterprise PDM from SolidWorks. Since that time, we have seen both company and product consolidation. UG Solutions acquired SDRC, which was in turn acquired by Siemens. Their respective products were combined to create TeamCenter. ComputerVision was acquired by PTC and their collective products were integrated to produce Windchill. Dassault Systèmes acquired both MatrixOne and SolidWorks. MatrixOne lives on as ENOVIA.

There is currently a multitude of PDM products available on the market. Their popularity is also steadily increasing mainly due to quicker and easier implementation. Some are offered by CAD vendors who have reinvented themselves as PLM software companies. This is meant to indicate that they provide not just design and manufacturing software products, but services and solutions that integrate product development into an enterprise. In most cases, a tight integration exists primarily for the CAD system developed by the same software provider. PDM systems currently available on the market include, among others, AutoDesk ProductStream, ENOVIA Smarteam, PTC Windchill, Siemens UGS TeamCenter, and SolidWorks Enterprise PDM. Table 6.1 depicts some PDM systems and the primarily supported CAD system offered by the same vendor. In addition to the PDM (or PLM) offered by CAD vendors, there are a number of popular PLM software systems developed by non-CAD vendors, such as SofTech, Arena, and so forth.

In general, systems offered by CAD vendors built their PDM with a strong connection to product data models created in CAD. Data sharing, such as BOM built upon the model tree

Table 6.1 Available PDM Systems and the Corresponding CAD Systems

PDM System	Corresponding CAD System
AutoDesk ProductStream	Autodesk Inventor
ENOVIA Smarteam	CATIA
PTC Windchill	Pro/ENGINEER
Siemens UGS TeamCenter	Siemens UGS NX
SolidWorks Enterprise PDM	SolidWorks

of the CAD models, is a natural approach. In addition, they facilitate engineering collaboration by taking advantage of the fully integrated CAD/CAE/CAM suite of the existing software product line. However, such systems are usually less flexible in terms of data integration or exchange with other engineering tools. On the other hand, systems offered by non-CAD vendors are more flexible and more general. For example, Arena Cloud PLM was developed in support of general engineering products without tying it with any specific mechanical CAD software. However, such software often requires more effort in creating product data. It offers strong data management capabilities but relies on external engineering capabilities for product design.

6.4.1 SYSTEMS OFFERED BY CAD VENDORS

In this subsection, we briefly mention prominent commercial PDM systems offered by CAD vendors, including AutoDesk ProductStream, ENOVIA Smarteam, Windchill, and Enterprise PDM.

6.4.1.1 AutoDesk® ProductStream® of Autodesk Inventor

AutoDesk® ProductStream® is the major software module that supports a design team in organizing, managing, and automating key design and release management processes (AutoCAD, 2009). With this software, the design can be reviewed and approved before releasing it to manufacturing. The software stores and manages work-in-progress design data and related documents with data management tools for workgroups (see sample screen in Figure 6.12(a)). Team members can accelerate development cycles and increase their company's return on investment in design data by driving design reuse. In addition, AutoCAD offers Balloons and BOM, which use standards-based balloons and part lists (see Figure 6.12(b)), and automatically update the BOM to seamlessly track any changes, which helps to keep teams on schedule by reducing costly breaks in production due to incorrect part counts, identification, and ordering.

6.4.1.2 ENOVIA Smarteam of CATIA

Dassault Systèmes' solutions for PLM include two product categories: ENOVIA and Smarteam. ENOVIA solutions include PDM, intellectual property life cycle management, virtual product design, collaboration solutions, and configured digital mock-up. Smarteam for life cycle and PDM enhances and accelerates the proliferation of product knowledge and business processes across the enterprise and product value chain with tighter CAD integrations.

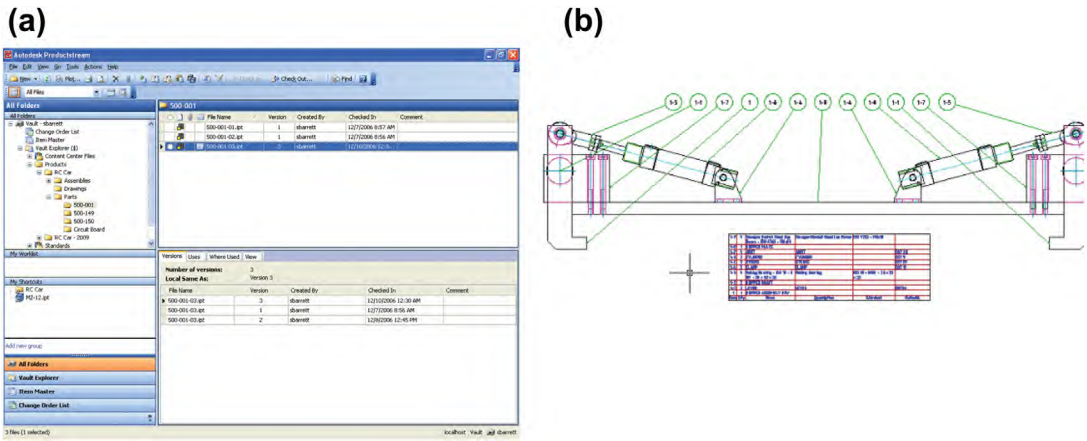


FIGURE 6.12

AutoDesk® ProductStream®. (a) Sample screen shot, and (b) BOM shown in standards-based balloons and part lists. (Figure courtesy of http://203.53.66.237/Data/Attachments/AutoCAD%20Mechanical%202009%20Detail_HR.pdf (AutoCAD, 2009).)

One of the key advantages of ENOVIA Smarteam is that the system provides collaborative offerings focused on product development processes supporting design, engineering, and enterprise activities (Dassault Systèmes, 2013). Design Collaboration enables dispersed design teams to work in collaboration in a single or Multi-CAD environment, to innovate new products and to reuse existing ones for faster time-to-market. Engineering Collaboration seamlessly combines cross-functional engineering-based activities throughout the product life cycle into a unified environment for effective data management and sharing among various organizational roles. Enterprise Collaboration provides a PLM solution throughout and across the extended enterprise, including the value chain. Supply Chain allows companies to leverage supply chain capabilities throughout the product life cycle and make their suppliers an integral part of product development.

6.4.1.3 Windchill by PTC

Windchill from PTC is considered to be one of the most advanced PDM systems available. It combines the power of client–server technologies with the implementation, manageability, and usability benefits of the web. It provides complete support for managing and communicating information about product structures and changes throughout the product life cycle. Windchill is built from scratch using all modern web technologies, and it is fundamentally based upon standard Internet, web, Java, and Oracle technologies at all levels of its architecture. Hence, Windchill claims their product to be "web-centric" as opposed to other "web-enabled" products.

6.4.1.4 TeamCenter by Siemens UGS NX

TeamCenter connects team members throughout the life cycle with a single source of product and process knowledge. TeamCenter's comprehensive portfolio of end-to-end PLM solutions gives users the flexibility to choose the right mix of solutions for product development needs.

TeamCenter's engineering process management solution allows users to integrate a company's global engineering teams by bringing together the product designs from all sites within a single PDM system. Team members can capture, manage, and synchronize product design data, then facilitate engineering change, validation, and approval processes. TeamCenter supports NX, Pro/ENGINEER, CATIA V5, AutoCAD, Inventor, Solid Edge, and SolidWorks.

6.4.1.5 SolidWorks Enterprise PDM

SolidWorks Enterprise PDM (EPDM) provides an easy way for designers to collaborate on product designs without worrying about version control or data loss. It stores CAD models and supporting documents in an indexed central repository that tracks versions and automates workflows to eliminate wasteful repetition. SolidWorks Enterprise PDM simplifies the process of managing design changes while improving product reuse by integrating within SolidWorks, AutoCAD, Autodesk Inventor, Pro/ENGINEER, and Windows Explorer.

6.4.2 SYSTEMS OFFERED BY NON-CAD VENDORS

In this subsection, we briefly mention commercial PDM systems that are not offered by CAD vendors, including SofTech ProductCenter® PLM (www.softech.com) and Arena Cloud PLM (www.arenasolutions.com).

6.4.2.1 SofTech ProductCenter® PLM

ProductCenter (sample screen capture shown in Figure 6.13) is a commercial software product that is an integrated suite of PLM software for managing product data. The software was engineered for the Microsoft Windows and UNIX operating systems. Along with core applications, it includes localized and web-based services. ProductCenter is suited for managing various types of CAD/CAE/CAM data, but it can be used for many forms of data management and product management. ProductCenter makes the use of spreadsheets for BOM management obsolete and provides organization for parts with various part types and attributes; in addition, all information managed can be accessed through the ProductCenter Hierarchy Explorer. This feature helps to facilitate small to mid-size manufacturers with a way to centralize product data, control the engineering change process, and share BOMs with suppliers. ProductCenter can be integrated with other CAD/CAE/CAM tools to help ease the management of product data from design to manufacturing.

6.4.2.2 Arena Cloud PLM

Arena pioneered cloud PLM applications. The company's products, including BOMControl, PartsList, and PDXViewer, enable engineering and manufacturing teams and their extended supply chains to speed up prototyping, reduce scrap, and streamline supply chain management. Arena cloud PLM applications simplify the BOMs and change management for companies of all sizes, and they offer the right balance of flexibility and control at every point in the product lifecycle—from prototype to full-scale production. These cloud-based applications enable manufacturers to manage BOMs, engineering change orders, product data exchange (PDX), and other key manufacturing files securely and efficiently.

PLM in the cloud is an internet-based system for managing a product and its associated information from concept to end of life. PLM in the cloud is growing in popularity with manufacturers around the

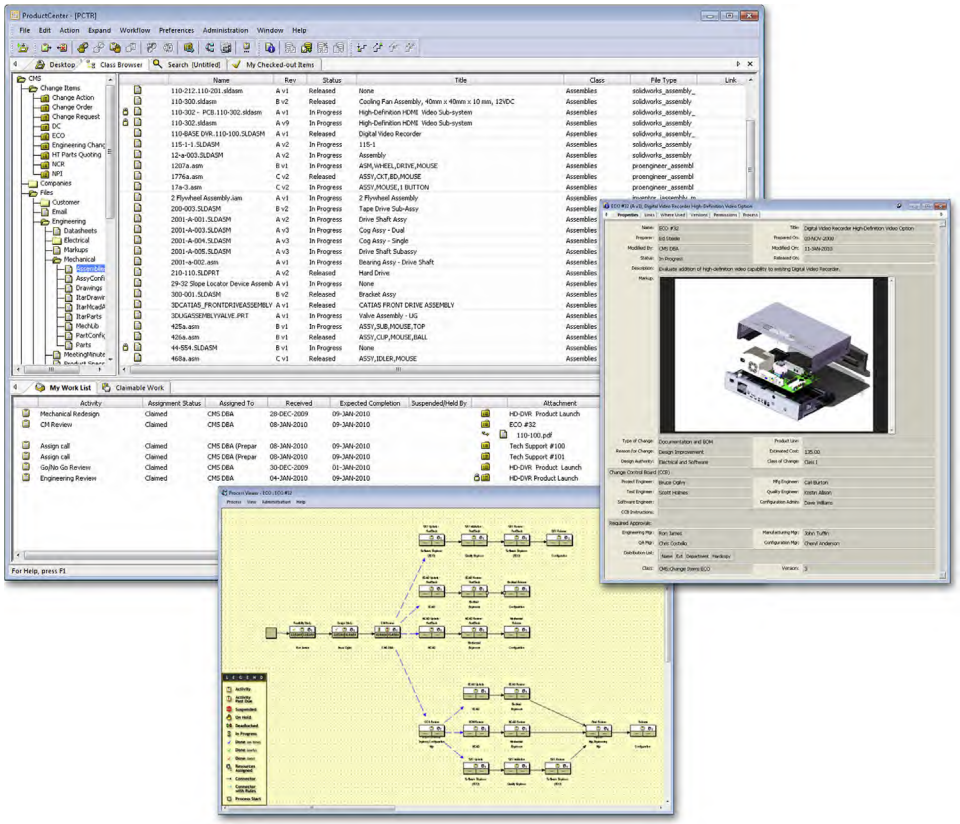


FIGURE 6.13

Sample screen shots of ProductCenter (en.wikipedia.org/wiki/ProductCenter).

world as a way to manage the stages of product development in order to collaborate, track, and regulate changes to the product. BOMControl (see Figure 6.14) keeps BOM data centralized, controlled, and up-to-date, resulting in fewer errors, less scrap and rework, higher quality, and better cost control.

6.5 PRODUCT DATA EXCHANGE

The frequent needs from product data exchange (PDX) encountered in product development involve solid model translations between CAD software systems, for both part and assembly levels. In general, for part model translation, the established approach both in theory and in practice is geometric data exchange (GDE) and feature data exchange (FDE). In GDE, the boundary representation (B-rep) of the object is translated from a source to a target CAD system (Spitzy and Rappoport, 2004). The resulting part in the target CAD system is lumped into one single entity—that is, one single solid feature in the model tree of the target CAD system, often called dumb geometry. Individual solid features and parametric data of the solid model created in the source CAD system are lost in the translation. On the

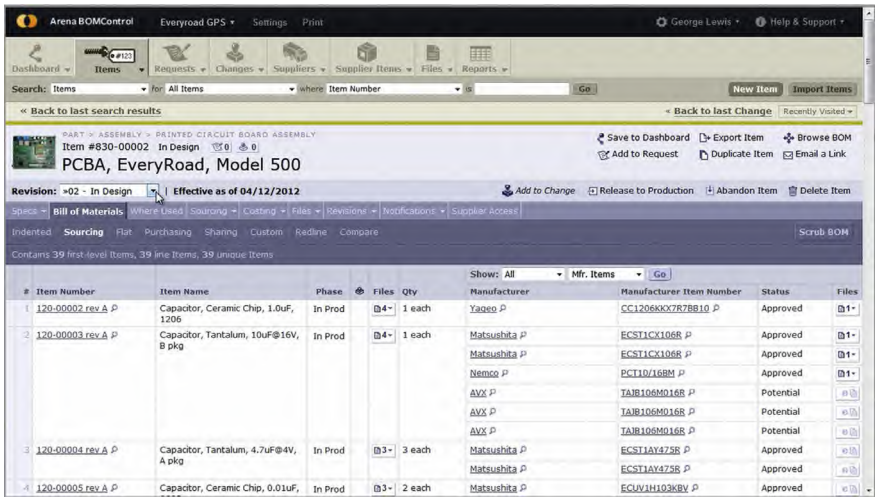


FIGURE 6.14

Sample screen shots of Arena cloud product life cycle management (en.wikipedia.org/wiki/ProductCenter).

other hand, in FDE, given a parametric history graph (or model tree) in a source system, the goal is to construct a graph in the target system that results in similar geometry while preserving as much parametric information as possible. FDE retains design intelligence and allows modifications at the receiving. However, it is not always technically possible to successfully exchange every feature. In the context of e-Design, FDE is much more desirable than GDE when a design change is anticipated for the CAD models being translated.

In assembly model translation, which is generally more involved than part translation, mating constraints defined using geometric entities of solid features must be faithfully retained from the source to target CAD systems. For example, a concentric mating constraint is often defined by selecting an outer surface of a cylinder (e.g., on an extrude feature of the mating part) and an inner surface of a hole (e.g., an extrude cut feature of the base part). As a result, translating an assembly model, in which its constituent parts are translated using GDE approach, is fundamentally deficient because feature information is not retained in translation. Even if features and parametric information is retained for the constituent parts, assembly model translation may not be as successful as expected. This is because FDE only results in similar geometry while preserving as much parametric information as possible, implying that feature information may be altered or incomplete. Therefore, in this section, we mainly focus on part solid model translation and only briefly mention assembly model translations using simple examples. More details about part and assembly model translations can be found in tutorial lessons of Projects S1 and P1 for SolidWorks and Pro/ENGINEER, respectively.

In this section, we start by introducing the viable options in support of CAD model translations in Section 6.5.1. In Section 6.5.2, we discuss direct model translations; both part and assembly examples are included. In Section 6.5.3, we discuss data exchange using neutral formats, including two important standards for part translation—IGES and STEP. In Section 6.5.4, we briefly mention several

third-party model translation software tools. In Section 6.5.5, we discuss a newly developed technology, solid feature recognition, which offers a better alternative for FDE.

6.5.1 DATA EXCHANGE OPTIONS

In general, there are three practical options of translating data from one CAD system to another: direct model translation, neutral file exchange, and third-party translators, as illustrated in Figure 6.15, in which System A is called the source and System B is called the target system.

Major CAD systems, such as SolidWorks, Pro/ENGINEER, NX, Unigraphics, and CATIA, directly read and/or write other CAD formats, simply by using File Open and File Save As options (Figure 6.15(a)). Because most CAD file formats and geometric modeling kernels are proprietary, this option is limited to selected CAD systems.

Another common method of translation is via an intermediate neutral format, as illustrated in Figure 6.15(b). The source CAD system exports out to this format and the target CAD system reads in this format and converts data into its native form. Some formats are independent of the CAD vendors, being defined by standard organizations, such as IGES (IGES, 2001) and STEP (Nell, 2001). Others, such as VDA (en.wikipedia.org/wiki/VDA_6.1), although owned by a company, are widely used and are regarded as quasi-industry standards.

There are a number of companies that specialize in CAD data translations and provide software that can read one system and write the information in another CAD system format (Figure 6.15(c)).

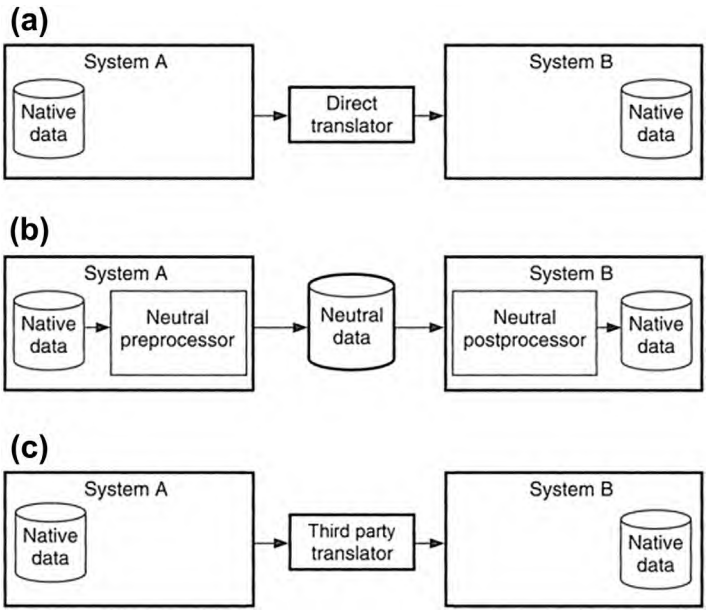


FIGURE 6.15

Three methods of data exchange. (a) Direct model translation, (b) neutral file exchange, and (c) third-party translators (Stokes, 1995).

These systems have their own proprietary intermediate format, some of which will allow reviewing the data during translation. Some of these translators work as stand-alone systems, whereas others require one or both of the CAD systems to be installed on the translation machine, as they use code such as application protocol interfaces (APIs) from these systems to read and write the data.

Because each CAD system has its own method of describing geometry, both mathematically and structurally, there is always some loss of information when translating data from one CAD system to another. The intermediate file formats are also limited in what they can describe, and they can be interpreted differently by both the source and target systems. It is therefore important in translating data between systems to identify what needs to be translated.

If the geometric model is required for the downstream process without anticipated design changes, then only the geometric description of the model needs to be translated. However, there are levels of detail. For example, are the data wireframe, surface, or solid sufficient? If a design change is anticipated, the feature information and model tree must be preserved between systems. In addition to geometric information, retaining the assembly structure may be required. In general, different data translation is required for different engineering activities.

From an e-Design perspective, GDE is generally sufficient, except for parts that anticipate changes, in which feature and parametric data must be available. When a fully integrated suite of CAD/CAE/CAM is not available or if engineering capabilities offered by certain software tools are not adequate, model translation is unavoidable. For example, to support motion analysis, geometry and coordinate systems of a solid part and subassembly are sufficient to support accurate calculation of mass properties and kinematic joint locations. To support finite element analysis, accurate solid or surface models of the respective CAD models are usually sufficient for finite element mesh generation. For CNC toolpath generation, solid part in CAD is directly useful. For some cases, even a surface model that represents the design surface (the part surface to which machining takes place) is sufficient, such as when using Mastercam. In general, GDE is sufficient to support CAE and CAM activities.

6.5.2 DIRECT MODEL TRANSLATIONS

As the engineering capabilities offered by major CAD systems progresses, CAD models can be translated to and from more CAD systems. In order to support model translations, a target CAD system must be able to open the model file of the source system, parse data stored in the native format of the source system, interpret the data, and map data entities to convert them into the format of the target system.

In this subsection, we offer a more in-depth discussion on the subject by importing parts and assembly created in Pro/ENGINEER to SolidWorks, as well as presenting examples of importing SolidWorks parts and assembly to Pro/ENGINEER.

6.5.2.1 Importing Pro/ENGINEER Parts to SolidWorks

SolidWorks offers two options for importing CAD models: importing solid features and importing geometry. We use the gear housing shown in Figure 6.16(a) as an example to illustrate both options. As shown in the Pro/ENGINEER model tree of Figure 6.16(a), there are eight datum features and 14 solid features. SolidWorks will try to import these 14 solid features from Pro/ENGINEER.

Using the option of importing solid features, SolidWorks translates 12 out of 14 features. The converted model and features listed in the browser are shown in Figure 6.16(b). As shown in

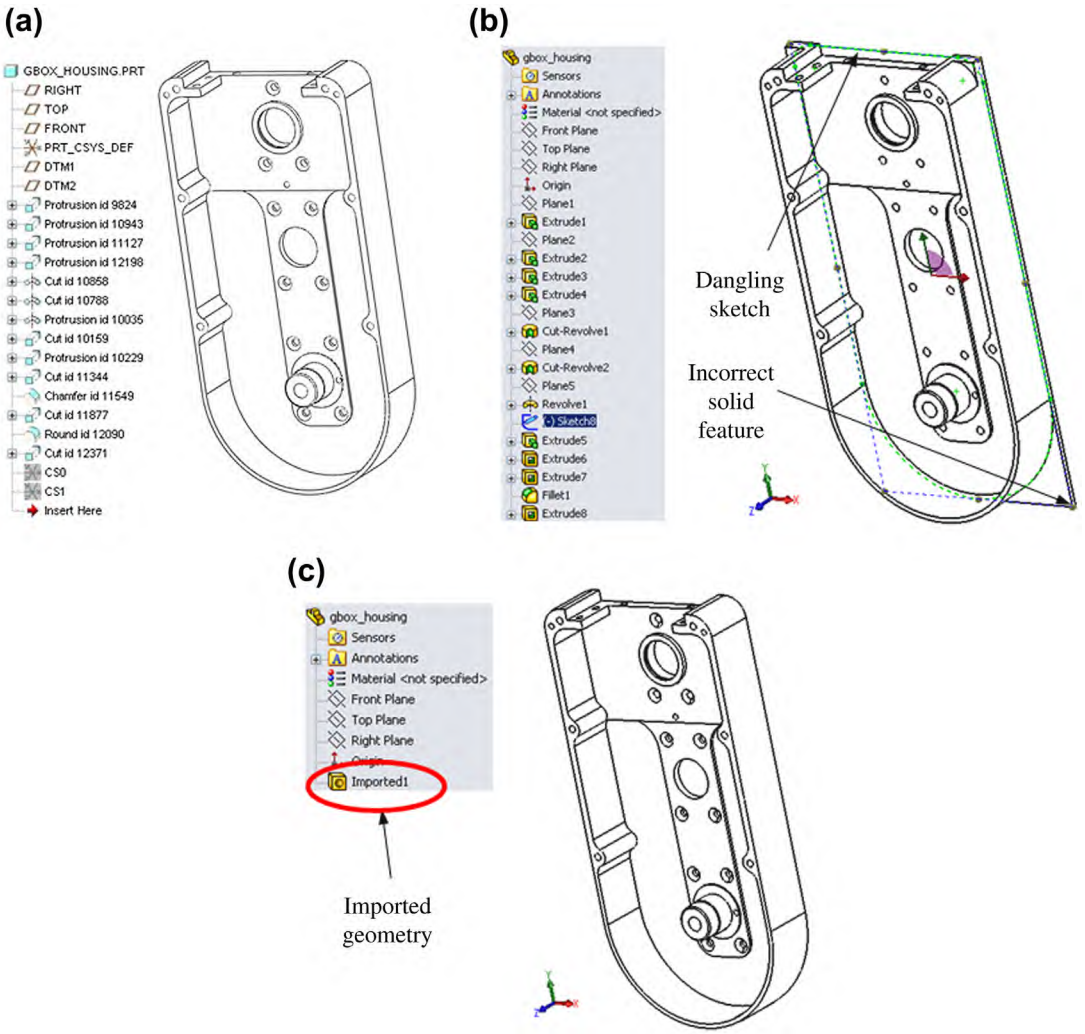


FIGURE 6.16

Part import from Pro/ENGINEER to SolidWorks. (a) Gear housing part in Pro/ENGINEER, (b) the translated solid model and features in SolidWorks using option of importing solid features, and (c) the translated solid model in SolidWorks using option of importing geometry.

Figure 6.16(b), there is one dangling sketch, *Sketch8*, representing the unrecognizable solid feature in addition to the chamfer feature. In addition, the back plate (*Extrude1* in the browser) is translated incorrectly. In general, SolidWorks is capable of importing some parts correctly and completely, especially when the solid features are relatively simple (but apparently not this gear housing part).

If we take a closer look at any of the solid features translated, such as *Extrude3*, the sketches (e.g., *Sketch3* of *Extrude3*) of the solid features do not have complete dimensions. A (−) symbol is placed in front of the sketch, indicating that the sketch is not fully defined.

Apparently, this translation is not satisfactory. Unfortunately, this translation represents a typical situation you will encounter for a large majority of the parts. In many cases, it may take a lesser effort to repair or recreate wrongly translated or unrecognized solid features. However, when you translate an assembly with many parts, the repairing effort could be substantial.

Importing parts using the option of importing geometry is more straightforward, which has a higher successful probability than that of importing solid features. The model is imported as a single entity *Imported1*; a dumb geometry appeared in the browser (see Figure 6.16(c)). As mentioned earlier, there is no parametric solid feature with dimensions and sketch imported. However, the geometry imported seems to be accurate. All the geometric features in Pro/ENGINEER were included in this imported feature. This translation is considered successful. If we do not anticipate making any change to the gear housing, this imported part is satisfactory.

6.5.2.2 Importing Pro/ENGINEER Assembly to SolidWorks

We import the input gear assembly shown in Figure 6.17(a) using both options. As shown in the left of Figure 6.17(a) (Pro/ENGINEER model tree), there are 11 parts in this assembly.

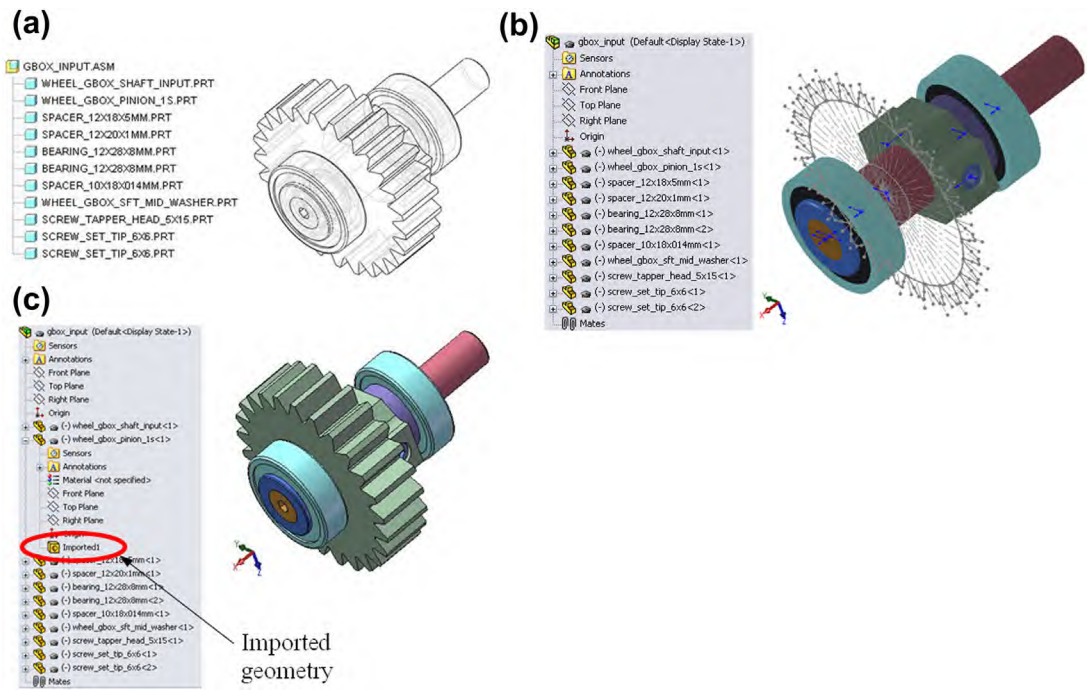


FIGURE 6.17

Assembly import from Pro/ENGINEER to SolidWorks. (a) Input gear assembly in Pro/ENGINEER, (b) the translated assembly in SolidWorks using option of importing solid features, and (c) the translated assembly in SolidWorks using option of importing geometry.

Using the option of importing solid features, parts are not completely imported, as shown in Figure 6.17(b). Major solid features are missing, such as pinion 1 (*wheel_gbox_pinion_1s<1>*), where most solid features are not imported. In fact, there are only two extrude features successfully imported. The remaining entities are mostly sketches. Some parts seem to be imported fine. However, the *Mates* branch in the browser is completely empty, implying that no assembly mates have been imported.

Apparently, this translation is not satisfactory. A nontrivial effort will have to be devoted to reconstructing the solid features (therefore, solid models) as well as the final assembly.

The option of importing geometry is also more straightforward for assembly. In fact, the assembly and all 11 parts seem to be correctly imported, as shown in Figure 6.17(c). By expanding any of the parts listed in the browser, such as the gear (*wheel_gbox_pinion_1s<1>*), we see an imported feature listed, as depicted in Figure 6.17(c). Again, there is no solid feature converted in any of the parts. In addition, the *Mates* branch is empty.

If we do not anticipate making any change to this input gear assembly, this imported assembly is satisfactory, except it does not have any assembly mates. Assembly of all 11 parts (maybe more, for other cases) will be a nontrivial effort. If you do not anticipate making changes in how these parts are assembled, you may merge all 11 parts into a single part, instead of assembling those using mating constraints.

A step-by-step detail of importing the Pro/ENGINEER part and assembly can be found in the tutorial lesson S1.3. You may go over the lesson to learn more about the model importing capabilities offered by SolidWorks.

6.5.2.3 Importing SolidWorks Parts to Pro/ENGINEER

The capabilities of importing SolidWorks models offered by Pro/ENGINEER are primitive. Only geometry data are imported without parametric feature information. We use a simpler crankshaft example shown in Figure 6.18(a) to illustrate the translations. As shown in the SolidWorks browser of Figure 6.18(a), there are three solid features, all boss-extrudes. After opening the SolidWorks model directly from within Pro/ENGINEER, the part is imported as one single feature (Imported Feature id 4), as shown in Figure 6.18(b). It is a model of dumb geometry. The part is not changeable.

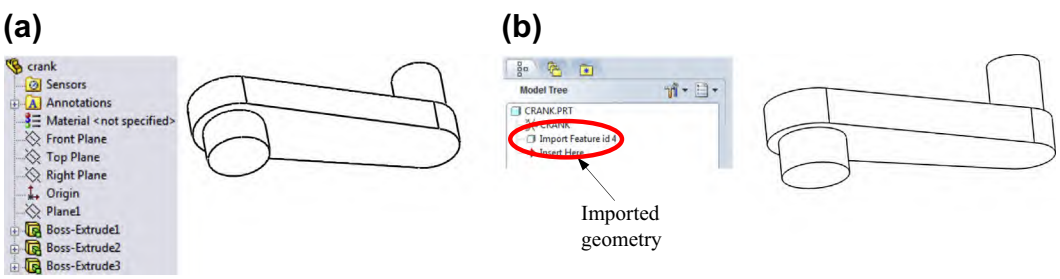


FIGURE 6.18

Part import from SolidWorks to Pro/ENGINEER. (a) Crankshaft part in SolidWorks, and (b) the imported solid model in Pro/ENGINEER.

6.5.2.4 Importing SolidWorks Assembly to Pro/ENGINEER

We import a simple slider-crank assembly shown in Figure 6.19(a) from SolidWorks to Pro/ENGINEER. As shown in the SolidWorks browser (left of Figure 6.19(a)), there are four parts in this assembly. The parts are parametric and assembly mating constraints are properly defined. After opening the SolidWorks assembly directly from within Pro/ENGINEER, the assembly is imported as four parts, as shown in Figure 6.19(b). The assembly and parts seem to be fine. However, no mating constraints are imported properly. As a result, dragging a part (e.g., the piston) leads to a disassembled model shown in the lower half of Figure 6.19(b). In addition, individual parts are imported as models of dumb geometry.

It is apparent that the capabilities offered by Pro/ENGINEER in importing SolidWorks models are not desirable. More details about the step-by-step process of importing the SolidWorks part and assembly discussed can be found in the tutorial lesson P1.3.

6.5.2.5 Data Exchange Between CAD and CAE/CAM

In addition to direct model translation between CAD systems, some CAE and CAM software reads CAD native files directly. For example, ANSYS reads CATIA and Pro/ENGINEER files, in addition to IGES, NX, SAT, and Parasolid. Mastercam reads AutoCAD, Pro/ENGINEER, Rhino, SolidWorks, Unigraphics, and CATIA, in addition to IGES, STEP, SpaceClaim, ACIS, Parasolid, and VDA. The success rate of importing native CAD models into CAE and CAM software is generally very good because the translation mostly involves parts only and requires only geometric data; in general no parametric features are involved.

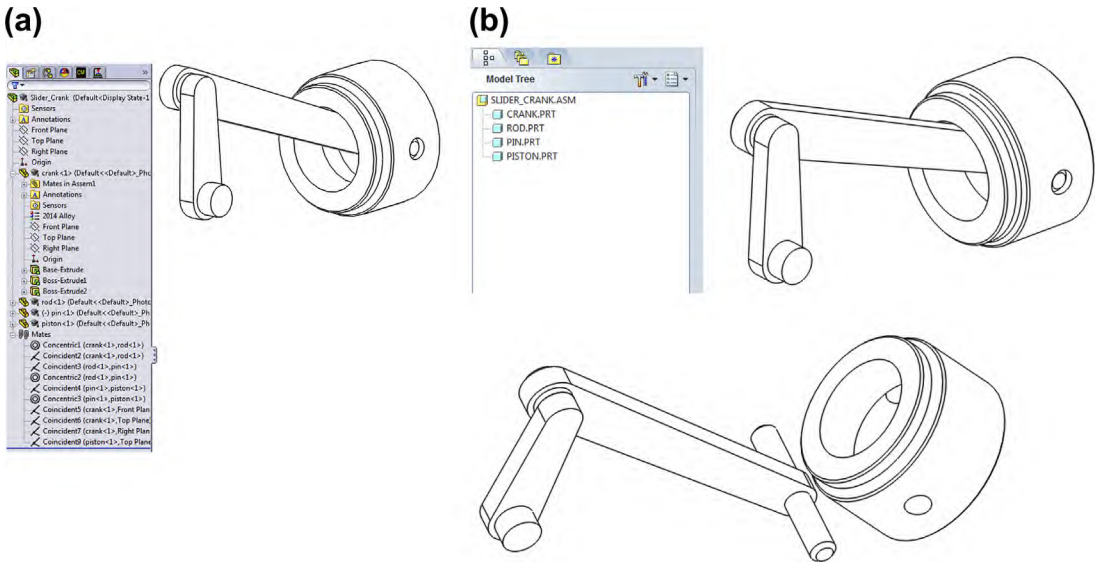


FIGURE 6.19

Assembly import from Pro/ENGINEER to SolidWorks. (a) Slider-crank assembly in SolidWorks, and (b) the translated assembly in Pro/ENGINEER.

6.5.3 NEUTRAL FILE EXCHANGE

The most commonly employed neutral files for CAD model translations are IGES and STEP application protocols (APs). In addition, several geometric kernels, such as ACIS (www.spatial.com) and Parasolid (www.eds.com/products/plm/parasolid), are becoming popular in serving as neutral formats for CAD model translations. Other formats commonly supported, such as STL and VRML, simplify true geometric data into faceted boundary representation for different purposes. Moreover, a data exchange file (DXF, en.wikipedia.org/wiki/AutoCAD_DXF) is the de facto format for drawing conversion, but less in support of solid model translation. In this subsection, we briefly introduce IGES and STEP APs with more detailed file formats and data structure included in appendices. Note that practically none of the neutral files are capable of supporting data exchange for parametric solid features.

6.5.3.1 IGES

The IGES (Initial Graphics Exchange Specification) project was started in 1979 by a group of CAD users and vendors, including Boeing, General Electric, Xerox, ComputerVision, and Applicon, with the support of the National Bureau of Standards (now known as NIST) and the U.S. Department of Defense (DoD). Soon after, it was adapted and recognized by the American National Standard Institute as a standard tool format. Consequently, IGES has become an acceptable and widely used neutral format for translator development by many CAD/CAM software vendors. After the initial release of STEP (ISO 10303) in 1994, interest in further development of IGES declined, and Version 5.3 (1996) was the last published standard. IGES has been used in the automotive, aerospace, and shipbuilding industries. These part models may have to be used years after the vendor of the original design system has gone out of business. IGES files provide a way to access these data decades later. The structure of the IGES file, data format, and simple examples are provided in Appendix 6A for further reference.

6.5.3.2 STEP (ISO 10303)

The work with the ISO 10303 standard, informally called STandard for the Exchange of Product model data (STEP), was initiated in 1984 with the goal to standardize the exchange of product data between product life cycle systems. After 10 years of work, the first parts of the STEP were published in 1994. The standard is a comprehensive set of specifications, covering many different product types (electronic, electromechanical, mechanical, sheet metal, fiber composites, ships, architectural, furniture, etc.) and many life cycle phases (design, analysis, planning, manufacturing, etc.). Using STEP-supporting tools, data can be exchanged by converting them from the native format of the source CAD system to the neutral ISO 10303-11 format, also known as an EXPRESS schema. Then, the target system imports the schema and converts it to its own native format. The EXPRESS schema defines not only the data types but also relations and rules applying to them. This makes it possible for the target system to validate the schema. EXPRESS is a textual and graphical data modeling language included in the STEP standard.

The STEP format is organized as a series of documents (referred to as "parts" in STEP terminology), with each part published separately. There are currently six series of STEP parts. The most important parts from an application perspective are the 200-series, also called APs, through which STEP meets the real world. The APs are the top parts, produced to meet specific data exchange requirements for a particular application. They cover a particular application and industry domain;

hence, they are most relevant for users of STEP. AP202, AP203, and AP214 have reached the status of an International Standard (IS) version. Other APs include AP202DIS, AP209DIS and AP214DIS (Draft International Standard), and AP214 CD II (Committee Draft). A complete list of APs can be found in Appendix 6B.

Among the APs, AP203 (configuration-controlled 3D designs of mechanical parts and assemblies) and AP214 (core data for automotive mechanical design processes) are the most popular and widely supported for CAD data exchange. AP203 defines the geometry, topology, and configuration management data of solid models for mechanical parts and assemblies. This file type does not manage colors and layers. AP214 has everything an AP203 file includes, but adds colors, layers, and geometric dimensioning and tolerance. AP214 is considered an extension of AP203 by many users.

It is worth noting that the initial release of ISO 10303 was aimed entirely at the exchange of explicit models, essentially a B-rep model, defined in terms of geometry, possibly with additional topological information providing connectivity relationships between geometric entities. This is because the state of the art was B-rep during the mid-1980s, when STEP development commenced. A B-rep model provides a complete representation of a solid shape, but retains no details of how that shape was created. As a result, no model tree and parametric feature information is retained using STEP for model translation. ISO 10303-108 on parameterization and constraints for explicit geometric product models (ISO, 2005) is a new STEP resource providing representations of parameters, explicit constraints, and explicit 2D sketches or profiles. More about STEP parts and APs can be found in Appendix 6B.

6.5.4 THIRD-PARTY TRANSLATORS

Neutral formats such as IGES and STEP support part geometry translation well. Third-party translators focus on feature and parametric data translation as well as assembly. In this subsection, we briefly mention two software tools that offer better solutions to CAD model translation: Proficiency (www.transcendata.com/products/proficiency) and TransMagic (www.transmagic.com/products/features).

6.5.4.1 Proficiency

Proficiency is a feature-based translation solution developed by International TechneGroup Inc. (www.iti-oh.com), headquartered in Milford, OH. Proficiency enables the transfer of design intelligence between major CAD systems, such as geometry, features, sketches, manufacturing information, metadata, assembly information, and drawings in the conversion process. Accurate and usable models are achieved with up to 95% automation, as claimed by the software vendor.

6.5.4.2 TransMagic

TransMagic offers translators that convert CAD files from one native file format to another. During the translation, TransMagic performs "geometry mapping," mapping from one CAD kernel to another. TransMagic avoids what are known as "stitching errors" by repairing geometry via techniques such as correcting slightly overlapping or misaligned surfaces, removing duplicate control points, and duplicate vertices. To minimize translation errors, TransMagic typically—but not always—translates directly from one native CAD kernel to another. Still, "stitching errors" (gaps and overlaps) can occur while trying to import the file and reinterpret geometry. TransMagic is available as a stand-alone program. It is also available as a plug-in for many CAD programs so that the Open and Save dialog boxes are extended with TransMagic's functionality.

6.5.5 SOLID FEATURE RECOGNITION

Feature recognition (FR) has been an active research topic for decades. Earlier study focused on recognizing manufacturing features, such as pockets, holes, slots, and so forth, created in CAD solid models (Shah, 1995). This effort led to capabilities implemented in several commercial CAM software tools, such as CAMWorks that automatically recognizes manufacturing features in SolidWorks models.

In the context of product data exchange, FR refers to recognizing geometric features embedded in a solid model of dumb geometry. This is called solid (or geometric) FR. The geometric model can be an IGES, STEP, or STL models exported from another CAD system, with no parametric feature information.

Many methods have been proposed for solid FR from numerous source files, such as from STL (Sunil and Pande, 2008), STEP (Bhandarkar and Nagi, 2000), or a B-rep model (van der Velden et al., 2010). In this subsection, we include Venkataraman's FR algorithm (Venkataraman et al., 2001) to provide readers with a brief understanding of the underline technique that supports solid FR. This method is relevant because it was recently implemented in a number of CAD systems, including SolidWorks and CATIA, which are capable of recognizing basic features, such as extrude, revolve, and, more recently, sweep. This capability has been applied primarily for support of solid model translations between CAD systems with some success, in which not only geometric entities but also parametric features are translated.

Venkataraman's FR algorithm uses a simple four-step process: (1) simplify imported faces, (2) analyze faces for specific feature geometry, (3) remove recognized feature and update model, and (4) return to Step 2 until all features are recognized. The process is illustrated in Figure 6.20. The simplification step involves surface format conversion and face merging. For instance, several connected B-spline patches or triangular facets as in STL models with identical (or similar) surface normal vectors could be combined and represented as a single planar face. The next step is to match the simplified faces for geometry resembling a specific solid feature. That is, given a specific feature type, the algorithm searches for the surfaces that resemble geometry associated with that feature. For instance, a hole would be constructed from a base circle extruded a given length (Figure 6.20(a)), producing a cylindrical

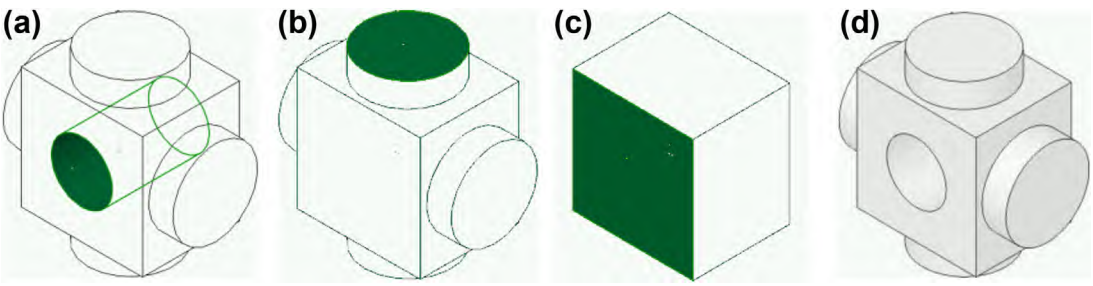


FIGURE 6.20

Illustration of Venkataraman's FR algorithm. (a) Imported surface model with hole surface selected, (b) hole recognized and removed, with extruded face of cylinder selected, (c) cylindrical extrusions recognized, with base block extrusion face selected, and (d) all features recognized and mapped to solid model (Chang, 2012).

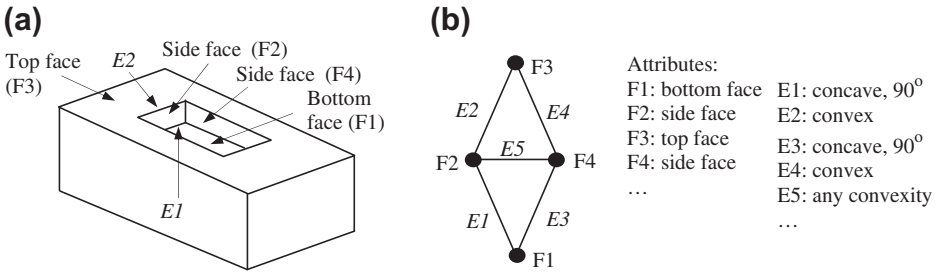


FIGURE 6.21

Feature grammars for the class of simple blind pocket. (a) The blind pocket feature, and (b) face adjacent graph and attributes.

surface. In this case, the hole would be seen as a negative feature. If an enclosed cylindrical surface is not found, a failure error is returned. If the correct geometry is found, the algorithm records the dimensions embedded in the composing surfaces (i.e., radius and depth in this case) and continues to the next step. Once a feature is recognized, it is temporally removed from the model. Continuing with the hole example, the hole feature would be filled, and the model would be updated to reflect the removal (Figure 6.20(b)). The algorithm would then analyze the faces of the updated part (Step 2) and continue the recognition process for the remaining features in the model (Figure 6.20(c)). Once all possible features are recognized, they are mapped to a new solid model of the part (Figure 6.20(d)), which is parametric with a feature tree that defines the feature regeneration sequence.

To recognize each feature, Venkataraman's FR algorithm abstracts a B-rep model as an attributed face adjacency graph. The faces of the feature are represented as nodes of the graph, while the edges in the feature are represented as lines that connect nodes of the graph. In addition, attributes are added to nodes and lines representing the topological and geometric characteristics of the corresponding faces and edges. For example, the blind pocket feature shown in Figure 6.21(a) consists of top, bottom, and side faces, as well as numerous edges. The partial face adjacency graph and attributes (e.g., edge convexity) are shown in Figure 6.21(b). As a result, the problem of FR becomes a subgraph detection problem in which the feature graph is matched to similar instances in a predefined feature library. This is done by a graph matching algorithm following prescribed rules and grammars.

One of the potential issues revealed in commercial FR software is design intent recovery. For example, the flange of a tubing would be created as a single revolve feature, where a sketch is revolved about an axis (Figure 6.22(a)). However, current FR implementations are not flexible. As shown in Figure 6.22(b), without adequate user interaction, the single sketch flange may be recognized as four or more separate features. Although the final solid parts are physically the same, their defining parameters are not. Such a batch mode implementation is not desired in recovering meaningful design intents.

In tutorial lesson S1.4, we use a housing example, an imported part shown in the left of Figure 6.23(a), to illustrate the steps of FR using FeatureWorks of SolidWorks. The FeatureWorks module of SolidWorks recognizes solid features on an imported object in a SolidWorks part document. Recognized features are (almost) the same as features that are created using parametric feature-based CAD software. Designers may edit the definition of recognized features and change their attributes and dimension parameters. For example, the fillet radius is changed from 0.0625 to 0.15 in, as shown on the

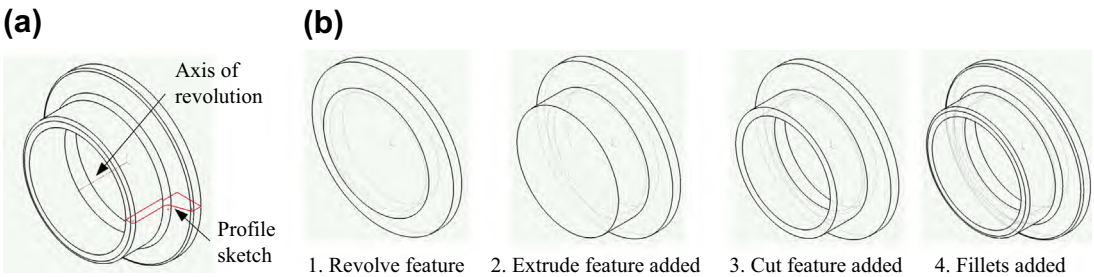


FIGURE 6.22

Feature recognition for a tubing flange. (a) A single revolved feature, and (b) four features: revolve, extrude, cut, and fillet (Chang, 2012).

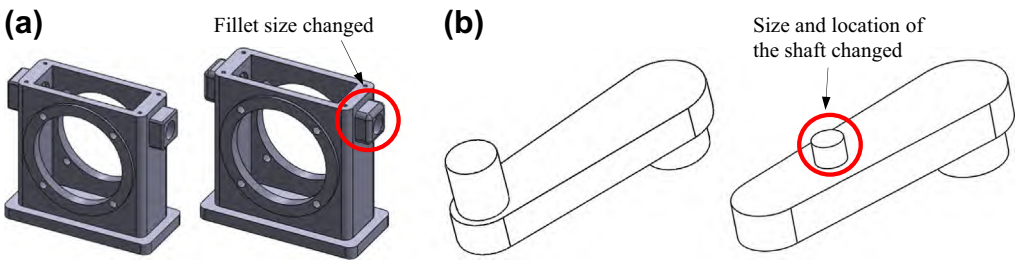


FIGURE 6.23

Examples for solid feature recognition. (a) The housing example employed for tutorial lesson S1.4, and (b) the crankshaft example employed for tutorial lesson P1.3.

right of Figure 6.23(a). For features that are based on sketches, designers can edit the sketches to change the geometry of the features. We introduce both the automatic and interactive options. Overall, one possible strategy for using FeatureWorks in solid FR is to use automatic FR to recognize as many features as possible, and then recognize the remaining features interactively.

Also in tutorial lesson P1.3, we use a very simple example—a crankshaft, as shown in Figure 6.23(b)—to illustrate the steps of FR using Pro/ENGINEER. Note that the FR capability implemented in Pro/ENGINEER is primitive and far less useful than that of FeatureWorks. Two shafts (extrusion features) are successfully recognized. For example, the size and location of one of the shafts are changed, as shown on the right of Figure 6.23(b). However, the crank body (the first extrusion feature) cannot be recognized because it is considered as the base feature, which is not recognizable by Pro/ENGINEER.

6.6 CASE STUDIES

Three case studies are presented in this section: SolidWorks Workgroup PDM, integrated testbed using Windchill, and infrastructure of tool integration for e-Design. The case study of SolidWorks PDM offers engineering students a quick overview about a viable solution for their needs in using PDM for

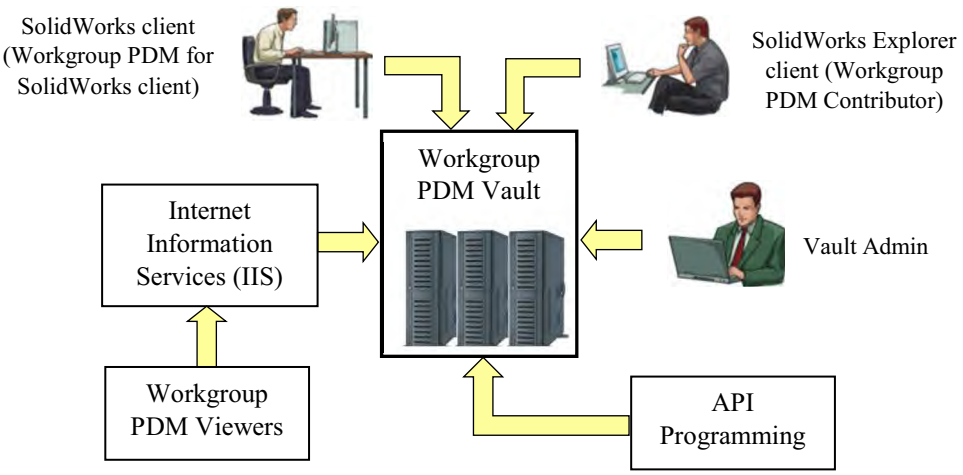


FIGURE 6.24

SolidWorks workgroup PDM structure.

support of design projects. The second study presents a case in which Windchill was integrated to support an engineering team working on reverse engineering projects. This case study is intended to show readers one possible scenario of using a commercial PDM system to streamline engineering activities. The third case study illustrates the concept and implementation of a software infrastructure for tool integration that supports e-Design.

6.6.1 SOLIDWORKS WORKGROUP PDM

There are two PDM systems offered by SolidWorks: SolidWorks Workgroup PDM and SolidWorks Enterprise PDM. Workgroup PDM is part of the SolidWorks Premium or Professional version and is considered a mid-range PDM system that is intended for small engineering workgroups, usually <10 team members. Enterprise PDM offers more robust capabilities and is better suited for larger teams. Because Workgroup PDM is part of SolidWorks, which many students may have access to, we offer a brief discussion of the Workgroup PDM as a case study.

The Workgroup PDM application is PDM software that runs inside the SolidWorks environment or as a stand-alone application inside SolidWorks Explorer. Workgroup PDM controls projects with procedures for check-in, check-out, revision control, and other administration tasks. A Workgroup PDM structure is illustrated in Figure 6.24, in which a "vault" sits at the center.

Workgroup PDM is very simple to install; installation can be completed in a few minutes by a nonexpert. During installation, a vault location is defined. The vault can be on a local drive or on a network server. In addition to the vault, user and administrator client software is also installed. The PDM license is managed by the SolidWorks license server, so no additional license configuration is required provided that SolidWorks is already installed. To support a project team, a global vault may be located on a network server, with carefully restricted administrative rights. Only one vault can be installed on a given computer.

Workgroup PDM associates metadata with CAD documents, Metadata is a series of text files that contain server options (user information, revision schemes, etc.) and file information (revision history,

owner, etc.). Workgroup PDM allows management and viewing of metadata associated with both local documents and documents in the vault.

When a file in the vault is opened and checked out, the file is downloaded from the vault to a local folder. If an assembly or drawing is opened, all of the referenced files are downloaded as well. Workgroup PDM compares the documents on the local drive to the documents in the vault, giving the user the option to overwrite previous versions with the latest versions from the vault. A file can be opened only by its owner. If the owner wishes to release a file for someone else to work on, he or she releases ownership. This file can then be checked out by others. Because Workgroup PDM is a mid-range PDM system best suited to small workgroups, it is easy to administer and use for a small team.

6.6.2 INTEGRATED TESTBED USING WINDCHILL

In this case study, we present an integrated testbed that supports defense logistics centers to conduct reverse engineering of aging systems and components (Chang et al., 2006). This testbed, which was constructed using commercial off-the-shelf software and equipment, supports three major engineering tasks: the reverse engineering that supports recovering of technical data from worn sample parts, reengineering that alters design for better performance or lesser cost, and fast prototyping that incorporates advanced manufacturing technologies to produce a functional or physical prototype of the part in small quantity in a short turnaround time.

Most reverse engineering solutions involve multidisciplinary design activities. Consequently, design collaboration is essential for a typical reverse engineering project to allow designers of different disciplines to perform their roles. In the integrated system, the design collaboration is based on two kinds of designers' interactions: asynchronous and synchronous. Asynchronous interactions involve email, notification, forums, and sharing documents where the designer is not required to respond in real-time. During synchronous interactions, the designer is required to respond in real-time. These synchronous interactions include whiteboard, chat room, model viewer, and video and audio communication. To meet these requirements, the integrated testbed supports the following:

- Appropriate distribution of activities to members of the team;
- Tools that can support real-time collaboration among team members with engineering information;
- An environment that organizes and provides easy access to engineering and other information related to the project for the team;
- A knowledge base that includes information related to different reverse engineering processes, tools, and techniques;
- A reverse engineering template that can be modified to support different reverse engineering processes and reduce the initial effort to set up products.

The testbed is intended to provide a software environment that supports multiple geographically dispersed designers. This principle extends to all reverse engineering activities, data, and collaborative activities, as well as to the infrastructure design. The testbed is set up using simple client–server architecture. The Windchill and communication module is housed in the server and is connected to the Internet. Multiple clients (users) access product and reverse engineering information from the servers using a web browser. Some product management functions supported by the servers are: (1) managing the product data and model in a structure through which a designer can easily locate the product data; (2) keeping the data secure and restrict illegal operations through basic file access

controls; (3) providing functions to manage the file operation privileges based on designers' roles in the team; and (4) supporting file status control to prevent the file inconsistency which may occur when two users modify the same file simultaneously.

To support real-time collaboration, a web-based tool has been developed (Figure 6.25). This collaborative tool supports text messaging, audio, video, sketching, and viewing of 3D models in real-time to facilitate activities required for meetings. To enhance collaboration among different members of the team, the collaborative 3D model viewer allows users to have a real-time synchronous view of the model, add notes to the 3D model, and exchange text and audio information in real-time. Collaborative meetings, if needed, can be scheduled in an ad-hoc manner. When a meeting

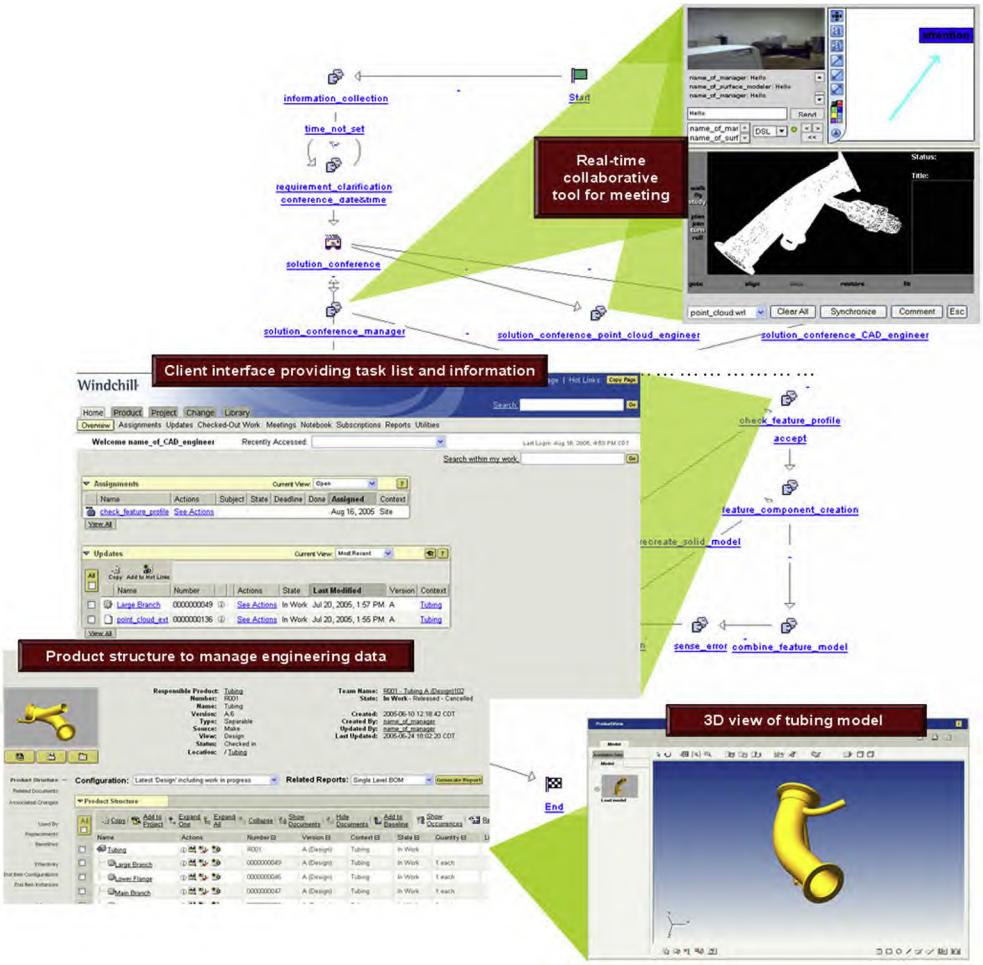


FIGURE 6.25

Reverse engineering template and activities in the integration testbed (Chang et al., 2006).

is scheduled, appropriate group members are sent an email that has the web link to the collaborative tool and the scheduled meeting time. During the scheduled time, all group members can log into the collaborative tool to discuss issues related to the project using the testbed.

In order to demonstrate the testbed, a case scenario was created. The reverse and reengineering scenario highlights (1) a systematic reverse engineering approach, (2) an enhanced ability of team member collaboration, and (3) a customized Windchill product management system. The reverse engineering of an airplane anti-icing tubing scenario involves an engineering team consisting of four members who are geographically distributed: manager, CAD engineer, and two point-cloud engineers. A template with a flow of activities (see Figure 6.25), along with appropriate instructions, has been set up in the Windchill environment. This template is the starting point for the manager to initiate a reverse engineering project. The initial steps for the manager involve gathering information, design constraints, and point-cloud information for the product. Once the information has been gathered, the manager creates the team and calls a meeting in the integration framework using the real-time collaborative tools (Figure 6.25) to discuss details of the project. After the meeting, the appropriate reverse engineering process can be selected and modified according to the requirements and needs of the project. The integration framework then supports accomplishing these tasks by appropriate users. Information and instruction on how to complete the different tasks are also available to the users from the testbed. Information created from each activity is uploaded in the testbed for other members of the team to view, access, evaluate, and use. These data are organized in a set of defined folders that follow the product structure to reduce the effort of finding the files. The progress of the project can be monitored by any member of the team at any given time. After each task is completed, the testbed sends appropriate notification to relevant team members to proceed to the next steps.

6.6.3 TOOL INTEGRATION FOR e-DESIGN

In this case study, we discuss an integration infrastructure that supports tool integration for e-Design (Tsai et al., 1995). The infrastructure supports engineers in creating CAD and simulation models of the mechanical system, accessing CAE tools to perform multidisciplinary engineering analyses, using planning tools to create and manage design processes, communicating and exchanging engineering data, conducting design trade-off analyses, and making informed decisions to yield a robust optimum design.

The infrastructure was designed to correlate various simulation models with a common product representation derived from a CAD model (see Figure 6.26). A base definition is created from the CAD model to serve as the common ground for design data sharing and collaboration. Engineering views are then derived from the base definition to support additional analysis requirements in each engineering discipline. Engineering view models are correlated, or mapped, with the base definition to support design collaboration and can be shared among the design team. Engineering tool wrappers provide service to their respective tools, including accessing the analysis model from the global database, converting model data into tool-specific data formats, transmitting data to a specified location, and retrieving and displaying analysis results. Finally, to create an e-Design tool environment, design process management has to be employed to define, disseminate, coordinate, and track design activities.

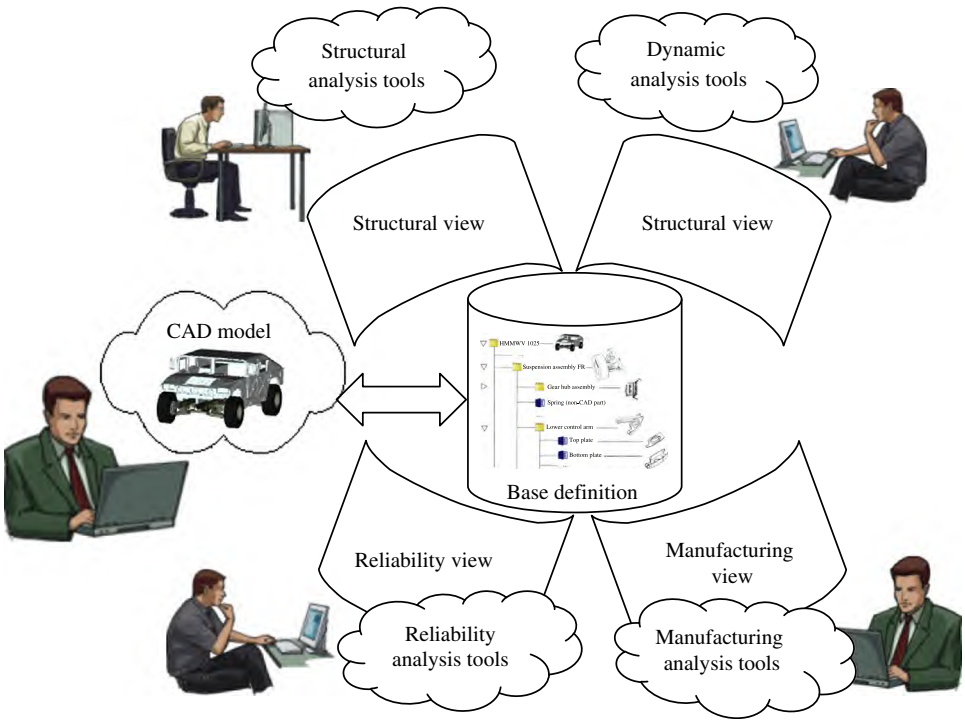


FIGURE 6.26

Concept of the software infrastructure for support of tool integration for e-Design (Tsai et al., 1995).

6.6.3.1 CAD and Base Definition

To support multidisciplinary CAE analyses, a base definition has to be built as the common ground among the CAE team members. The base definition contains two major types of information: an entity hierarchy and entity attributes. The entity hierarchy describes how the components of the system are grouped together. The hierarchy of an engine example is depicted in Figure 6.27. If an entity in the hierarchy is an assembly, it can be expanded to display its components or collapsed without showing its components. The entity attributes for a part include mass, center of gravity (CG), moments of inertia, material properties, and geometry information. The default coordinate system defined in the CAD model is used as the local coordinate system for the part; the CG is reported relative to it. Geometry information of the mechanical system is kept in the original CAD format and later transferred to different formats to support various simulation model creations. In addition, parameters used to build the CAD geometry need to be extracted and later used to support design trade-offs. Attribute information for an assembly differs from that for a part, with the addition of assembly information describing the position and orientation of individual components relative to a local reference frame. Once all the hierarchy information and assembly information are available, the global position and orientation of the individual part or assembly can be automatically calculated.

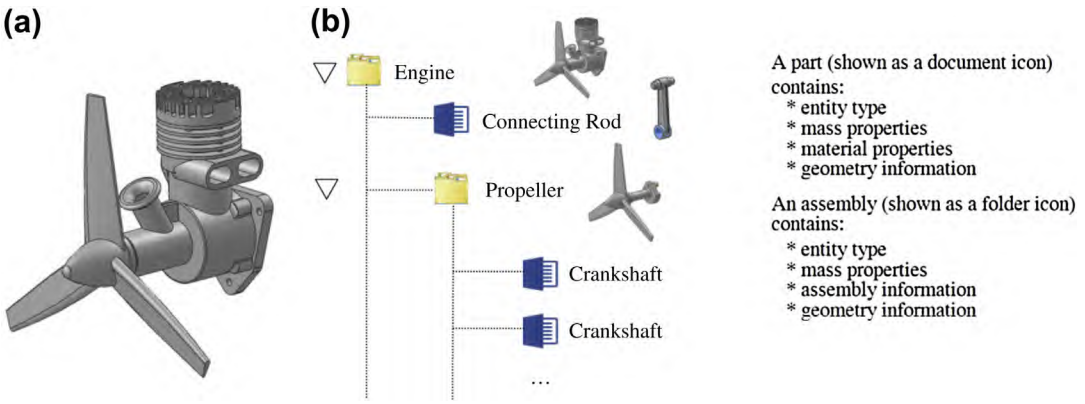


FIGURE 6.27

Engine example. (a) Computer-aided design model, and (b) product base definition (Tsai et al., 1995).

In the e-Design environment, design parameters are associated with the dimensions of features in the parameterized CAD models. The design parameters are considered to be attributes of entities in the base definition; they stay with the entities when they are regrouped in engineering views to create assemblies. The feature-based design parameters serve as a common language to support design trade-offs across various engineering disciplines where relevant performance of the mechanical systems is measured.

6.6.3.2 Disciplines and Views

In addition to establishment of a common base definition, the integration infrastructure has to support engineers from different disciplines (e.g. dynamics and structure) to create their own simulation models and to perform engineering analyses. Due to the fact that data requirements vary from discipline to discipline, the infrastructure has to allow engineers to augment the model data in the base definition with discipline-specific data. While allowing diverse data to be added, the infrastructure has also to maintain the consistency among these data so that the common ground is not broken and design trade-offs across different disciplines can still be performed.

To address these issues, a key concept—engineering views—is introduced in the infrastructure. The engineering views are associated with their corresponding disciplines to support the data augmentation in a natural way. Furthermore, the data created in the engineering views can be shared among the engineers that perform engineering analyses in the same discipline. Therefore, effort is saved in creating engineering models.

Another important function the engineering views need to support is maintaining a consistent product data set for the mechanical system being evaluated. The mappings between each view model and the base definition have to be established (see Figure 6.28). All the engineering models, along with their simulation results and the CAD model (brought in as the base definition), are correlated through these mappings, allowing meaningful communication among the CAE team members and design trade-offs across disciplines.

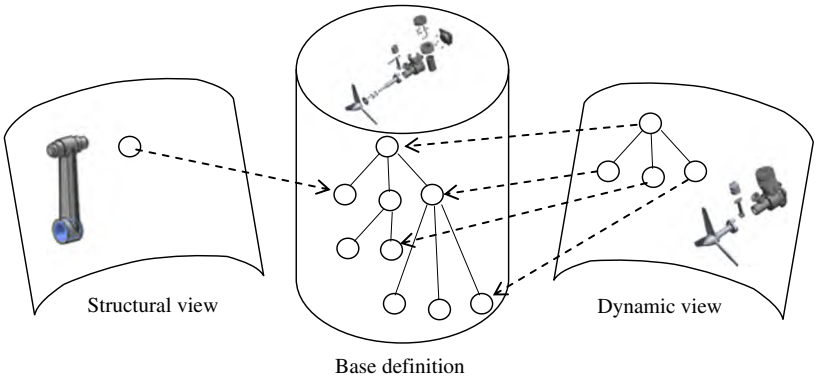


FIGURE 6.28

Mappings between base definition and engineering views (Tsai et al., 1995).

Another benefit of establishing these mappings is that they can be used as the foundation of automating the engineering model (re)creation that is required during design iterations. Once a design change is proposed, each engineering discipline has to re-evaluate the performance of the new design. However, if the engineering model has to be regenerated from scratch, then the effort previously spent in model creation is wasted. Therefore, a mechanism for retaining the mapping relationships between the engineering model and the base definition would greatly speed up the design cycle. As the CAD model is modified, the engineering model could be automatically modified with very little effort.

As an example, in the dynamics view, the assembly hierarchy defined in the CAD model might not be suitable for multibody mechanical system definition. For that, parts or assemblies need to be regrouped into bodies and then connection joints, allowing relative motion between bodies, need to be defined. Once the regrouping is performed, the composite mass, CG, moments of inertia, and assembly information of the body can be calculated automatically based on individual component mass properties and assembly information. The dynamic view of the engine example shown in Figure 6.27 is shown in Figure 6.29, in which parts under the engine assembly in CAD (and base definition) are grouped and mapped to dynamic view, consisting of three subassemblies and one part that correspond to the motion model of the example. Once the data required in the dynamics view are created, the tool wrapper can be invoked to export the mechanical system definition into the local working environment and create a motion simulation model (e.g., using DADS). The results of the dynamics analyses can also be retrieved via the tool wrapper to the global database to support other simulations.

6.6.3.3 Engineering Tool Wrappers

After the engineering view models are created in the e-Design environment, engineers are ready to perform various analyses using engineering tools. To interface with each engineering tool, the infrastructure needs to provide services to prepare the analysis model from the global database, translate it into the tool-specific data format, and transmit data to the dedicated location specified by the tool. Each wrapper provides services to a specific tool; in other words, it is customized to properly interface with that tool. For instance, the engineer can use the PATRAN wrapper to transfer the

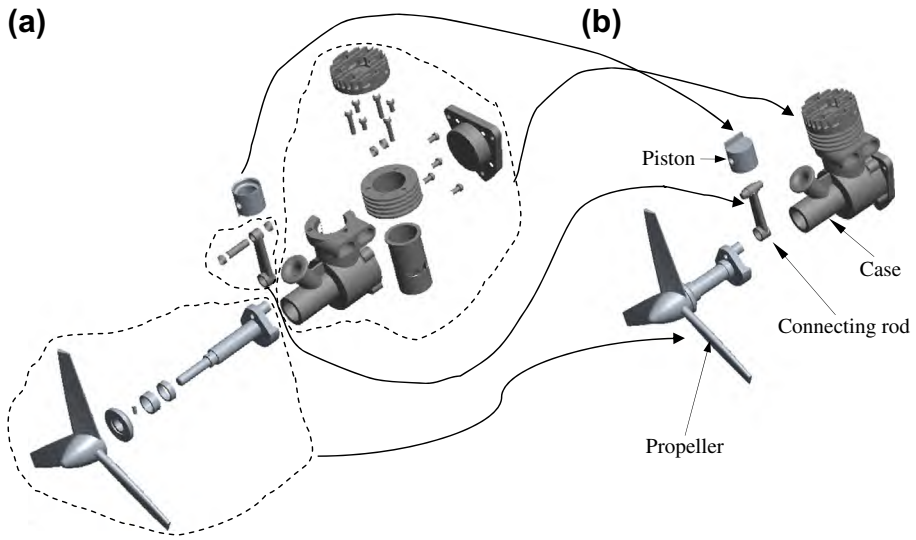


FIGURE 6.29

Dynamic view of the engine example. (a) Computed-aided design (CAD) assembly hierarchy in base definition, and (b) CAD model in exploded view showing four bodies for motion analysis.

PATRAN hyperpatch model from the structural view and visualize the dynamics simulation results to determine a peak load, which is required by the structural analysis tools. Another service that wrappers provide is retrieving analysis results from each tool. The analysis results are interpreted by the wrappers and then stored in the global database for later use.

6.6.3.4 Design Process Management

Effective utilization of the integrated e-Design environment and collaboration among a CAE team is contingent on a number of data generation and communication factors intrinsic to the operational requirements of the engineering disciplines and the product design process in general. As described above, the e-Design environment consists of a number of individual engineering views integrated via a common base definition and a suite of tool wrappers. Enhanced collaboration among engineers of this environment, vis-à-vis the design process management, will occur when engineers can identify data sources that meet their analysis requirements and communicate their input data needs and simulation results.

The design process management envisioned for the integrated e-Design environment employs process definition and analysis, task identification and dissemination, and progress tracking to provide enhanced collaboration among the CAE team. Process definition enables CAE teams to specify and capture data generation, design, analysis, and design trade-off activities in the e-Design process and represent data flow between process activities and perspectives. Process analysis allows the chief engineer to identify potential bottlenecks in a process and aids in the definition of an optimal design process. Process activities can be characterized by user data requirements, by operational parameters such as time and resource requirements, and by activity dependencies, thus providing information

supporting the determination of a project plan in compliance with the timeframe and resources specified for completion of the design project. The design project plan can then be displayed to all team members to provide them with an awareness of where data come from and where they go, thus defining responsibilities and obligations in the design project. Finally, progress tracking allows the CAE team to review and update the design project plan and to correlate the design project plan with the design process. By this approach, CAE team members will be provided with a frame of reference, with respect to project planning and environment operations, supporting communication and collaboration to achieve project objectives and adhere to project schedules and milestones.

6.6.3.5 Design Collaboration

Lastly, design collaboration includes the communication board, design process management, design parameterization, and design trade-off. The communication board provides a means for CAE team members to communicate about design tasks. The design parameterization module assists engineers in identifying design parameters to facilitate effective design evaluation. The design trade-off module collects performance evaluation information from the engineering tools and assists engineers in obtaining optimal design.

6.7 SUMMARY

In this chapter, we addressed two issues that are relevant to product design: PDM and product data exchange. We provided an overview on the practical means for file management and an introduction to PDM technology and systems. We presented fundamentals of PDM, including the data that PDM manages, the capabilities of PDM, and the benefits and successful stories of PDM technology. We also discuss commercially available PDM systems. For product data exchanges, we discussed the numerous approaches and translators available, as well as their strengths and limitations. We include examples to demonstrate the part and assembly model translations between Pro/ENGINEER and SolidWorks, as well as FR in both Pro/ENGINEER and SolidWorks. Finally, we offered case studies that illustrate the practical use of SolidWorks Workgroup PDM, a practical case of using PDM for support of reverse engineering projects, and infrastructure for support of tool integration for e-Design.

We discussed lots of topics and offered more diverse materials in this chapter. We hope this chapter is not too difficult for you to read and digest. After reading this chapter, you should have acquired basic knowledge and a good understanding of these two important issues in product design.

Together with the topics discussed in the previous chapters of this book, you should have a very good understanding of the various subjects involved in product data modeling, such as geometric modeling, CAD theories, mechanical assembly, design parameterization, and PDM. More importantly, we hope Part I of this book provided you with a fundamental understanding of product modeling principles and modern engineering tools for solid and assembly modeling, so that you can apply the principles and software tools to support practical design applications.

With a good understanding of product modeling, you should feel competent in using CAD tools in support of your design projects and move into other important topics for e-Design, such as product performance evaluation, product manufacturing and cost estimating, and design theory and methods, which are discussed in the remaining parts of this book.

APPENDIX 6A: IGES FILE STRUCTURE AND DATA FORMAT

Similar to most CAD systems, IGES is based on the concept of entities. Entities could range from simple geometric objects, such as points, lines, plane, and arcs, to more sophisticated entities, such as dimensions. Entities in IGES are divided into three categories:

1. Geometric entities such as arcs, lines, and points that define the object,
2. Annotation entities, such as dimensions and notes that aid in the documentation and visualization of the object,
3. Structure entities that define the associations between other entities in an IGES file.

An IGES file is a sequential file consisting of a sequence of records. The file formats treat the product definition to be exchanged as a file of entities, with each entity being represented in a standard format, to and from which the native representation of a specific CAD/CAM system can be mapped.

An IGES file consists of five sections, which must appear in the following order: Start section, Global section, Directory Entry (DE) section, Parameter Data (PD) section, and Terminate section, as shown in Figure 6A.1 (Stokes, 1995). In fact, an IGES file is composed of 80-character ASCII records. Each of the sections can be identified by the letters S, G, D, P, and T, respectively, appearing in the 73rd column of each record or line in the IGES file. The role of these sections is summarized in the following.

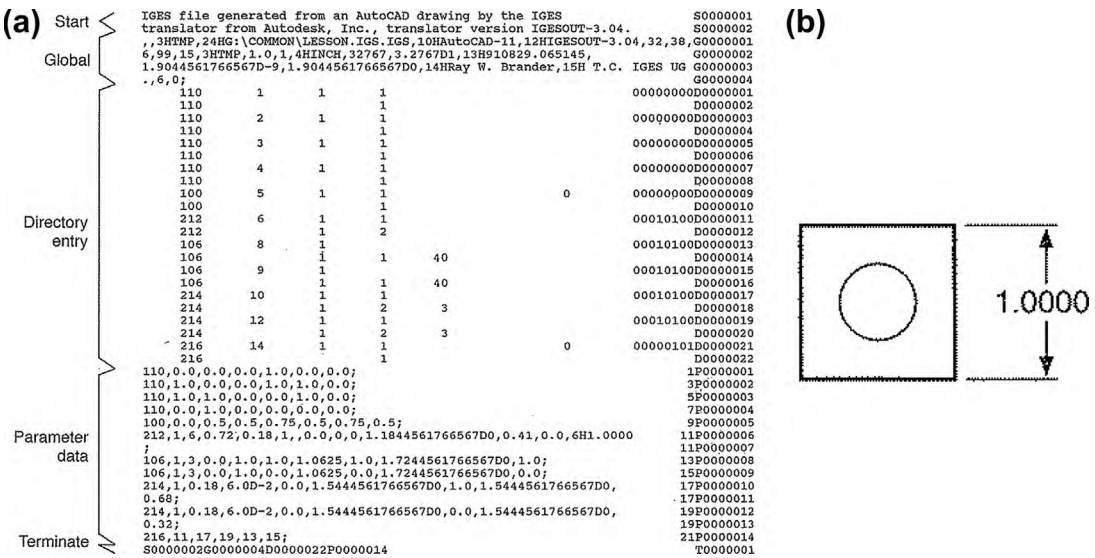


FIGURE 6A.1

Anatomy of a sample IGES file. (a) IGES file with sections labeled, and (b) equivalent graphics (Stokes, 1995).

START SECTION

The start section is for free-form text generated by the user or the CAD/CAM system to tell the receiver basic information about the IGES file, commonly described as a "prologue" to the IGES file. It is essentially a human-readable introduction to the file. This section contains information such as the names of the sending (source) and receiving (target) CAD/CAM systems and a brief description of the product being converted.

GLOBAL SECTION

The global section provides information that pertains to the entire file. It is a fairly short section, typically in three or four lines. This information describes the preprocessor and information needed by the postprocessor to interpret the file. Some of the parameters that are specified in this section are:

1. Characters used as delimiters between individual entries and between records (usually commas and semicolons respectively),
2. The name of the IGES file itself,
3. Vendor and software version of sending (source) system,
4. Number of significant digits in the representation of integers and single and double precision floating point numbers on the sending systems,
5. Date and time of file generation,
6. Model space scale,
7. Model units,
8. Minimum resolution and maximum coordinate values,
9. Name of the author of IGES file.

DIRECTORY ENTRY SECTION

The directory entry (DE) section is an index listing each entity in the file, together with certain attributes associated with them. The entry for each entity occupies two 80-character records that are divided into a total of 20 eight-character fields as shown in Figure 6A.1. The first and the eleventh (beginning of the second record of any given entity) fields contain the entity type number such as 100 for circle, 110 for lines, etc. The second field contains a pointer to the parameter data entry for the entity in the PD section. The pointer of an entity is simply its sequence number in the DE section. Some of the entity attributes specified in this section are line font, layer number, transformation matrix, line weight, and color. A list of IGES entities is provided in Table 6A.1 for reference.

PARAMETER DATA SECTION

The parameter data (PD) section contains the actual data defining each entity listed in the DE section. For example, a straight line entity is defined by the six coordinates of its two endpoints. Although each entity always has two records in the DE section, the number of records required for each entity in the PD section varies from one entity to another (the minimum is one record) and depends on the amount of data. Parameter data are placed in free format in columns 1–64. The parameter delimiter (usually a comma) is used to separate parameters and the record delimiter (usually a semicolon) is used to

Table 6A.1 IGES Entities (Stokes, 1995)

IGES Entity Number	Form Numbers	IGES Entity Name
0		Null entry
100		Circular arc
102		Composite curve
104	0−3	Conic arc
106	1−63	Copious data
108	(−1)−1	Plane
110		Line
112		Parametric spline curve
114		Parametric spline surface
116		Point
118	0−1	Ruled surface
120		Surface of revolution
122		Tabulated cylinder
123		Direction (G)
124	0−12	Transformation matrix
125	0−4	Flash
126	0−5	Rational B-spline curve
128	0−9	Rational B-spline surface
130		Offset curve
132		Connect point
134		Node
136		Finite element (G)
138		Nodal displacement and rotation
140		Offset surface
141		Boundary (G)
142		Curve on a parametric surface
143		Bounded surface (G)
144		Trimmed surface
146		Nodal results (G)
148		Elements results (G)
150		Block
152		Right angular wedge
154		Right circular cylinder
156		Right circular cone frustum
158		Sphere
160		Torus
162	0−1	Solid of revolution
164		Solid of linear extrusion
168		Ellipsoid
180		Boolean tree
182		Selected component (G)

Continued

Table 6A.1 IGES Entities (Stokes, 1995) *Continued*

IGES Entity Number	Form Numbers	IGES Entity Name
184		Solid assembly
186		Manifold solid B-rep object (G)
190		Plane surface (G)
192		Right circular cylindrical surface (G)
194		Right circular conical surface (G)
196		Spherical surface (G)
198		Toroidal surface (G)
202		Angular dimension
204		Curve dimension (G)
206		Diameter dimension
208		Flag note
210		General label
212	0−105	General note (F, G)
213		New general note (G)
214	1−12	Leader (arrow)
216	0−2	Linear dimension (G)
218	0−1	Ordinate dimension (G)
220		Point dimension
222	0−1	Radius dimension (G)
228	0−3	General symbol (G)
230	0−1	Sectioned area (G)
302		Associativity definition
304	1−2	Line font definition
306		Macro (G)
308		Subfigure definition
310		Text font definition
312	0−1	Text display template
314		Color definition
316		Units data (G)
320		Network subfigure definition
322	0−2	Attribute table definition
402	1−21	Associativity instance (F, G)
404	0−1	Drawing (G)
406	1−31	Property (F, G)
408		Singular subfigure instance
410	0−1	View (G)
412		Rectangular array subfigure instance
414		Circular array subfigure instance
416	0−4	External reference (G)
418		Nodal load/constraint
420		Network subfigure instance
422	0−1	Attribute table instance

Table 6A.1 IGES Entities (Stokes, 1995) *Continued*

IGES Entity Number	Form Numbers	IGES Entity Name
430		Solid instance
502		Vertex (G)
504		Edge (G)
508		Loop (G)
510		Face (G)
514		Shell (G)

Notes:
1. All information is based upon IGES version 5.1, September 1991.
2. F = Some or all forms of this entity have been obsoleted by newer entities.
3. G = Some or all forms of this entity have not been fully tested.

terminate the list of parameters. Both delimiters are specified in the Global section of the IGES file. Column 65 is left blank. Columns 66–72 on all PD records contain the entity pointer specified in the first record of the entity in the DE section.

TERMINATE SECTION

The Terminate section contains a single record that specifies the number of records in each of the four preceding sections for checking purposes.

Figure 6A.2 shows another IGES sample file containing only two POINT (Type 116), two CIRCULAR ARC (Type 100), and two LINE (Type 110) entities. It represents a slot, with the

(a)

```
                                                                        S      1
1H,,1H;,4HSLOT,37H$1$DUA2:[IGESLIB.BDRAFT.B2I]SLOT.IGS;,                G      1
17HBravo3 BravoDRAFT,31HBravo3->IGES V3.002 (02-Oct-87),32,38,6,38,15,  G      2
4HSLOT,1.,1,4HINCH,8,0.08,13H871006.192927,1.E-06,6.,                   G      3
31HD. A. Harrod, Tel. 313/995-6333,24HAPPLICON - Ann Arbor, MI,4,0;     G      4
     116       1       0       1       0       0       0       0       1D     1
     116       1       5       1       0                               0D     2
     116       2       0       1       0       0       0       0       1D     3
     116       1       5       1       0                               0D     4
     100       3       0       1       0       0       0       0       1D     5
     100       1       2       1       0                               0D     6
     100       4       0       1       0       0       0       0       1D     7
     100       1       2       1       0                               0D     8
     110       5       0       1       0       0       0       0       1D     9
     110       1       3       1       0                               0D    10
     110       6       0       1       0       0       0       0       1D    11
     110       1       3       1       0                               0D    12
116,0.,0.,0.,0,0,0;                                                     1P     1
116,5.,0.,0.,0,0,0;                                                     3P     2
100,0.,0.,0.,0.,1.,0.,-1.,0,0;                                          5P     3
100,0.,5.,0.,5.,-1.,5.,1.,0,0;                                          7P     4
110,0.,-1.,0.,5.,-1.,0.,0,0;                                            9P     5
110,0.,1.,0.,5.,1.,0.,0,0;                                             11P     6
S      1G      4D     12P      6                                        T      1
```

(b)

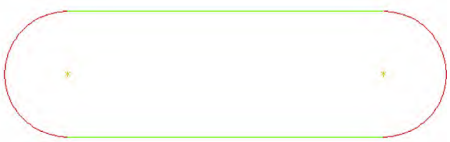


FIGURE 6A.2

Sample IGES file. (a) IGES file contents, and (b) equivalent graphics (en.wikipedia.org).

points at the centers of the two half-circles that form the ends of the slot, and the two lines that form the sides.

As stated earlier, the file is divided into five sections: Start, Global, Directory Entry, Parameter Data, and Terminate, indicated by the characters S, G, D, P, or T in column 73. The characteristics and geometric information for an entity are split between two sections—one is in a two-record, fixed-length format (the DE section), whereas the other is in a multiple-record, comma-delimited format (the PD section), as can be seen in a more human-readable representation of the file. When displayed, the user should see two yellow points, one located at the origin of model space [0,0,0], two red circular arcs, and two green lines.

For a more in-depth discussion on IGES, readers are referred to books such as Kennicott (1996).

APPENDIX 6B: STEP DATA STRUCTURE AND APPLICATIONS PROTOCOLS

STEP consists of several hundred documents called parts, as illustrated in Figure 6B.1 Every year new parts are added or new revisions of older parts are released. This makes STEP the biggest standard within ISO. Each part has its own scope and introduction. These parts are assigned a name and number and grouped together with common functions within a specific range.

The ten series parts comprise the computer-interpretable area of STEP. This area allows all users to operate by the same guidelines and rules necessary to maintain consistent, accurate data exchange. EXPRESS is the data modeling language used to make STEP computer-interpretable. The language can be compiled to produce "C" structures, SQL statements, or other similar types of information. This language is an important advantage of STEP over IGES, which offers nothing comparable.

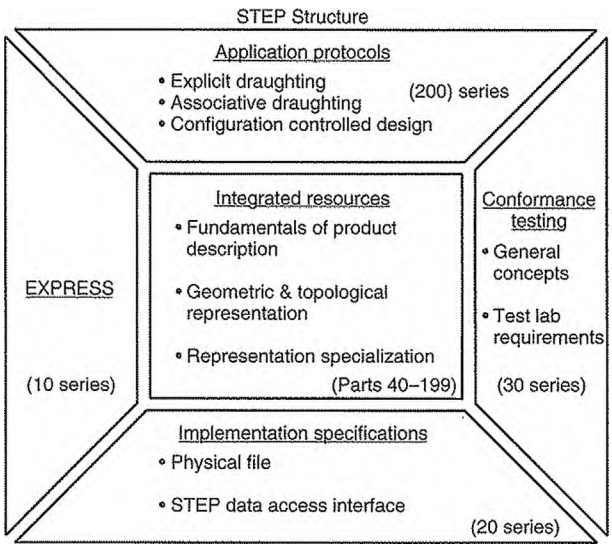


FIGURE 6B.1

The structure of STEP (Dincau, 1995).

The 20-series parts define the physical file and database-sharing exchange area and are the enabling tools for STEP data translation.

The 30-series parts define conformance testing requirements and are used for data and application verification.

The 40-series parts are considered to be the bread and butter of STEP. These parts contain such generic resource information as raw geometry and display attributes, among other things. These and the 100-series parts are the tools used to create application protocols (APs).

The 100-series parts are similar in concept to the 40-series parts in their use to create application protocols. The difference between the two is that the 100-series is specific to an application area.

The 200-series is where STEP meets the real world. Each AP includes a scope describing its purpose, an activity diagram describing the functions that an engineer needs to perform within that scope, and an application requirement model describing the information requirements of those activities. These information requirements are then mapped into the common set of integrated resources. The result is a data exchange standard for the activities within the scope.

The ultimate goal is for STEP to support the product life cycle, from conceptual design to final disposal, for all kinds of products. However, it will be a number of years before this goal is reached. The most tangible advantage of STEP to users today is the ability to exchange design data as solid models and assemblies of solid models. Other data exchange standards, such as the newer versions of IGES, also support the exchange of solid models, but not as well.

A list of the STEP APs is given in Table 6B.1. The ability to support many protocols within one framework is one of the key strengths of STEP. All the protocols are built on the same set of integrated resources, so they all use the same definitions for the same information. For example, AP203 and AP214 use the same definitions for three-dimensional geometry, assembly data, and basic product information. Therefore, CAD vendors can support both with one piece of code.

Table 6B.1 A List of STEP Application Protocols

Part	Description
201	Explicit drafting
202	Associative drafting
203	Configuration-controlled design
204	Mechanical design using boundary representation
205	Mechanical design using surface representation
206	Mechanical design using wireframe representation
207	Sheet metal dies and blocks
208	Life cycle product change process
209	Design through analysis of composite and metallic structures
210	Electronic printed circuit assembly, design, and manufacturing
211	Electronics test diagnostics and remanufacture
212	Electrotechnical plants
213	Numerical control process plans for machined parts
214	Core data for automotive mechanical design processes

Continued

Table 6B.1 A List of STEP Application Protocols *Continued*

Part	Description
215	Ship arrangement
216	Ship molded forms
217	Ship piping
218	Ship structures
219	Dimensional inspection process planning for CMMS
220	Printed circuit assembly manufacturing planning
221	Functional data and schematic representation for process plans
222	Design engineering to manufacturing for composite structures
223	Exchange of design and manufacturing DPD for composites
224	Mechanical product definition for process planning
225	Structural building elements using explicit shape rep
226	Shipbuilding mechanical systems
227	Plant spatial configuration
228	Building services
229	Design and manufacturing information for forged parts
230	Building structure frame steelwork
231	Process engineering data
232	Technical data packaging
233	Systems engineering data representation
234	Ship operational logs, records and messages
235	Materials information for products
236	Furniture product and project
237	Computational fluid dynamics
238	Integrated CNC machining
239	Product life cycle support
240	Process planning

QUESTIONS AND EXERCISES

6.1. When you open or save a CAD model using Pro/ENGINEER or SolidWorks (or other CAD systems), you may notice that the CAD software is able to open and save the model into many different formats. It is important for a CAD user and a designer to know these file formats, where they come from, and their use in solid modeling and product design. In this exercise, you are asked to create a list of file formats that are supported by either Pro/ENGINEER or SolidWorks (or the CAD system you have access to) and report the following:

a. File suffix and the name of the file format.

b. Source of the file format (for example, the CAD system that creates the file or a neutral format). For neutral format, please report the nature of the format, its use in data exchange, and pros and cons.

c. Provide the sources of your information, web links, technical report, etc.

6.2. Pick a commercial PDM (or PLM) system, carry out a case study on the system, and report the following:
- **a.** Brief information about the software company that develops and commercializes the PDM system.
- **b.** Major functions of the system, its strengths, and its weaknesses.
- **c.** Major companies who are using the system.
- **d.** Provide sources of your information, web links, technical report, etc.

6.3. In Section 6.3.4 we discussed two stories regarding the use of PDM systems in industry. Please conduct a similar study to report a similar successful story and report the following:
- **a.** Name of the company or organization, and the nature of its product or project.
- **b.** The PDM system the company uses.
- **c.** How did the company or organization use the PDM system? What was the driving factor that propelled the company to adopt PDM?
- **d.** What is the benefit that company is able to obtain (for example, reduced time-to-market, increased engineering efficiency, reduced prototype costs, reduced late changes to parts, etc.)? Provide quantitative data if possible.
- **e.** Provide the sources of your information, web links, technical report, etc.

REFERENCES

Abramovici, M., Sieg, O.C., 2002. Status and development trends of product lifecycle management systems. In: Proceedings of the IPPD 2002 Wroclaw. Wroclaw, Poland.

ACIS, Spatial Technology, Inc. www.spatial.com.

AutoCAD Mechanical, 2009. Autodesk, Inc, Sausalito, CA. 203.53.66.237/Data/Attachments/AutoCAD%20Mechanical%202009%20Detail_HR.pdf.

Bhandarkar, M.P., Nagi, R., 2000. STEP-based feature extraction from STEP geometry for Agile manufacturing. Computers in Industry 41, 3–24.

Buchal, R.O., July 24–26, 2006. The use of product data management (PDM) software to support student design projects. In: The 3rd CDEN/RCCI International Design Conference. Toronto, Ontario.

Caldwell, B., Mocko, G.M., August 3–6, 2008. Product data management in undergraduate education, DETC2008–50015. In: Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference. Brooklyn, New York, USA.

Chandrasegaran, S.K., Ramania, K., Sriram, R.D., Horváth, I., Bernard, A., Harik, R.F., Gao, W., 2013. The evolution, challenges, and future of knowledge representation in product design systems. Computer-Aided Design 45, 204–228.

Chang, K.H., March 2012. A review on shape engineering and design parameterization in reverse engineering, Reverse Engineering. InTech 268-4, 162–186. ISBN 979-953-307.

Chang, K.H., Choi, K.K., Wang, J., Tsai, C.S., Hardee, E., June 1998. A Multi-level product model for simulation-based design of mechanical systems. Concurrent Engineering Research and Applications (CERA) Journal 6 (2), 131–144.

Chang, K.H., Siddique, Z., Edke, M., Chen, Z., 2006. An integrated testbed for reverse engineering of aging systems and components. Computer-Aided Design and Applications 3 (1–4), 21–30.

CIMdata, 1998. Product Data Management: The Definition. An Introduction to Concepts, Benefits, and Terminology.

Crnkovic, I., Asklund, U., Persson Dahlqvist, A., 2003. Implementing and Integrating Product Data Management and Software Configuration Management. Artech House, London.

Dassault Systèmes, 2010. SolidWorks Corp., White Paper: PDM vs PLM: It All Starts with PDM.

Dassault Systèmes, 2013. Enovia V6R2013x Enovia Factsheet. www2.3ds.com/fileadmin/PRODUCTS/Enovia/PDF/Datasheets/Enovia-V6R2013x-factsheet.pdf.

Dincau, M.A., 1995. Solid modeling and product data exchange using step (PDES/STEP). In: LaCourse, D.E. (Ed.), Handbook of Solid Modeling. McGraw-Hill, Inc.

Hepplemann, J., 1998. PDM for the enterprise. Mechanical Engineering 120 (10), 74–79.

Horváth, I., 2004. A treatise on order in engineering design research. Research in Engineering Design 15, 155–181.

IGES, January 2001. In: Lide, D.L. (Ed.), A Century of Excellence in Measurements, Standards, and Technology—a Chronicle of Selected NBS/NIST Publications, 1901–2000. NIST Special Publication 958.

ISO, 2005. Industrial Automation Systems and Integration—Product Data Representation and Exchange—Part 108: Integrated Application Resource: Parameterization and Constraints for Explicit Geometric Product Models. 2005, International Organization for Standardization (ISO), Geneva, Switzerland. ISO 10303–108.

Kennicott, P.R., 1996. Initial Graphics Exchange Specification. IGES 5.3. U.S. Product Data Association.

Lee, S.G., Ma, Y.-S., Thimm, G.L., Verstraeten, J., 2008. Product lifecycle management in aviation maintenance, repair and overhaul. Computers in Industry 59, 296–303.

Lee, D., Shin, H., Choi, B.K., 2010. Mediator approach to direct workflow simulation. Simulation Modeling Practice and Theory 18 (5), 650–662.

Liu, D.T., Xu, X.W., 2001. A review of web-based product data management systems. Computers in Industry 44, 251–262.

Miller, E., 1998. PDM moves to the mainstream. Mechanical Engineering 120 (10), 74–79.

Nell, J., 2001. STEP on a Page, NIST. www.nist.gov/sc5/soap.

Owen, R., Horváth, I., 2002. Towards product-related knowledge asset warehousing in enterprises. In: Proceedings of the 4th International Symposium on Tools and Methods of Competitive Engineering. TMCE, pp. 155–170.

Parasolid, EDS, Corporate Headquarters, Plano, Texas: www.eds.com/products/plm/parasolid.

Qiu, Z.M., Wong, Y.S., June 2007. Dynamic workflow change in PDM systems 58 (5), 453–463.

Shah, J.J., 1995. Parametric and Feature-based CAD/CAM: Concepts, Techniques, and Applications. John Wiley & Sons.

Spitzy, S., Rappoport, A., 2004. Integrated feature-based and geometric CAD data exchange. In: Elber, G., Patrikalakis, N., Brunet, P. (Eds.), ACM Symposium on Solid Modeling and Applications.

Stark, J., 2011. Product Lifecycle Management: 21st Century Paradigm for Product Realization (Decision Engineering), second ed. Springer-Verlag. 10: 0857295454.

Stokes, H., 1995. Solid modeling and the initial graphics exchange specification (IGES). In: LaCourse, D.E. (Ed.), Handbook of Solid Modeling. McGraw-Hill, Inc.

Sunil, V.B., Pande, S.S., 2008. Automatic recognition of features from freeform surface CAD models. Computer-Aided Design 40, 502–517.

Tang, D., Qian, X., 2008. Product lifecycle management for automotive development focusing on supplier integration. Computers in Industry 59, 288–295.

Tsai, C.S., Chang, K.H., Wang, J., August 1995. Integration infrastructure for a simulation-based design environment. In: Proceedings of the Computers in Engineering Conference and the Engineering Data Symposium, ASME Design Theory and Methodology Conference, Boston, MA. pp. 9–20.

van der Velden, C., Zhang, H.-L., Yu, X., Jones, T., Fieldhouse, I., Bil, C., September 2010. Extracting engineering features from b-rep geometric models. In: 27th International Congress of the Aeronautical Sciences, ICAS2010, Nice France. pp. 19–24.

Venkataraman, S., Sohoni, M.A., Kulkarni, V., June 4–8, 2001. A graph-based framework for feature recognition. In: Sixth ACM Symposium on Solid Modeling and Applications. Michigan, Ann Arbor, pp. 194–205.

CHAPTER

STRUCTURAL ANALYSIS

7

CHAPTER OUTLINE

7.1 Introduction 327
7.2 Analytical Methods 328
7.2.1 Strength of Materials 328
7.2.2 Energy Method 330
7.2.3 Linear Elasticity 333
7.2.4 Failure Criteria 336
7.2.5 Uncertainties, Variations, and Safety Factors 338
7.3 Finite Element Methods 340
7.3.1 A Simple Example 341
7.3.2 Finite Element Formulation 345
7.3.3 *p*-Version FEA 351
7.3.4 The Meshless Method 356
7.3.5 Using Finite Element Method 358
7.4 Finite Element Modeling 359
7.4.1 General Process and Potential Pitfalls 360
7.4.2 Idealization and Simplification 361
7.4.3 Mesh Generation and Refinement 363
7.4.3.1 Automatic Mesh Generation 365
7.4.3.2 Semiautomatic Mesh Generation 369
7.4.4 CAD Model Translations 371
7.4.5 Loads and Boundary Conditions 372
7.4.6 Results Checking 373
7.4.7 Strategy for Complex Problems 376
7.5 Commercial FEA Software 376
7.5.1 General-Purpose Codes 377
7.5.2 Specialized Codes 378
7.6 Case Study and Tutorial Examples 378
7.6.1 Case Study 378
7.6.2 Tutorial Examples 380
7.6.2.1 Cantilever Beam 380
7.6.2.2 Thin-Walled Tube 382
7.7 Summary 384

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00007-7

Copyright © 2015 Elsevier Inc. All rights reserved.

Appendix 7A: The Default in.-lb$_m$-sec Unit System **384**
Questions and Exercises **385**
References **389**

Structural analysis is an essential part of product design. Mechanical components must be strong and durable so that the entire mechanical system is able to sustain operating loads for its intended operating conditions. The structural integrity of the entire system must be ensured.

In structural analysis, different types of problems are solved depending on the operating conditions and intended use of the product. These include static, buckling, vibration, transient dynamics, frequency responses, and others. Except for simple elastostatic problems, very few of these can be solved in closed form using analytical equations. Most, if not all, encountered in engineering design must rely on numerical methods for solutions, such as finite element methods. Although finite element methods (FEMs) are powerful, understanding theories and analytical methods is crucial for two reasons. One, the underlying mechanics and physics of the analytical methods help us understand how structures behave under specific conditions. Second, it is critical to verify numerical results obtained from methods such as FEM. Very often, it is possible to use analytical solutions to check FEA results for more complex problems. The bottom line is that a solid background in structural mechanics is essential to use FEM software correctly and effectively. As an analogy, one must know basic addition, subtraction, multiplication, and division before punching keys on a calculator.

In addition to analytical methods, in this chapter we will devote major discussion to finite element methods. Note that our focus will not be on presenting comprehensive FEM theory. Instead, we will focus on the discussion of essential elements in practical use of FEM software for modeling and analysis. The goal of this chapter is to help the reader become confident and competent in using FEM for creating adequate models and obtaining reasonably accurate results to support product design. Those who are interested in FEM theory may refer to excellent books, such as Pilky and Wunderlich (2002), Szabó and Babuška (1991), and Bathe (2007). Major FEM software packages, both general-purpose and specialized codes, will be briefly discussed to provide a general understanding of software availability and options as well as sufficient information for making reasonable choices. In addition, advanced methods based on the finite element concept, such as the meshless method, will be introduced briefly.

A human middle ear model is included as a case study, in which a mechanics study was carried out using FEA that explains how the ossicles work to eventually facilitate hearing aid development. In addition, two practice examples, a cantilever beam and a thin-walled tube modeled with Pro/MECHANICA® Structure and SolidWorks© Simulation, are offered. Detailed instructions on bringing up these models and steps for carrying out FEA are given in Projects P3 and S3.

Overall the objectives of this chapter are (1) to provide basic FEM theory using simple examples to explain how the method works, (2) to help the reader become familiar with FEM modeling and analysis to effectively use these tools for design, (3) to familiarize the reader with existing commercial software, and (4) to help the reader use Pro/MECHANICAL Structure and/or SolidWorks Simulations for basic applications after going through the tutorial lessons.

7.1 INTRODUCTION

Structural analysis comprises the set of mechanics theories that obey physical laws required to study and predict the behavior of structures. The subjects of structural analysis are engineering artifacts whose integrity is judged largely on their ability to withstand loads. Structural analysis incorporates the fields of mechanics and dynamics as well as the many failure theories. From a theoretical perspective, the primary goal of structural analysis is the computation of deformations, internal forces, and stresses. In practice, structural analysis reveals the structural performance of the engineering design and ensures the soundness of structural integrity in design without dependence on direct testing.

In the mechanical and aerospace industries, engineers often confront the challenge of designing mechanical systems and components that can sustain operating loads, meet functional requirements, and last longer. It is imperative that these structures contain a minimum of material to reduce cost and increase efficiency of the mechanical system, such as in terms of fuel consumption. The geometry of these load-bearing structural components is usually complicated because of strength and efficiency requirements. Three of such structural components are shown in Figure 7.1.

A number of approaches have been developed to support the design of structural components and systems, such as shape optimization (Chang and Edke, 2010), topology optimization (Bendsoe and Sigmund, 2003), and reliability-based design (Choi et al., 2007). Many have been employed to create designs for challenging applications, such as those shown in Figure 7.1. The success of these design methods is largely attributed to accurate underlying analysis methods that are built on fundamental mechanics theory and physics laws.

To perform an accurate analysis a structural engineer must determine such information as structural loads, geometry, support conditions, and materials properties. The results of his or her analysis typically include support reactions, stresses, and displacements. This information is then compared to criteria that indicate the conditions of failure. Advanced structural analysis may examine dynamic response, stability, and nonlinear behavior.

Analytical methods make use of analytical formulations that apply mostly to simple linear elastic models, lead to closed-form solutions, and can often be solved by hand. Such methods include strength of materials, energy methods, and linear elasticity. Numerical approaches, such as FEM, are more

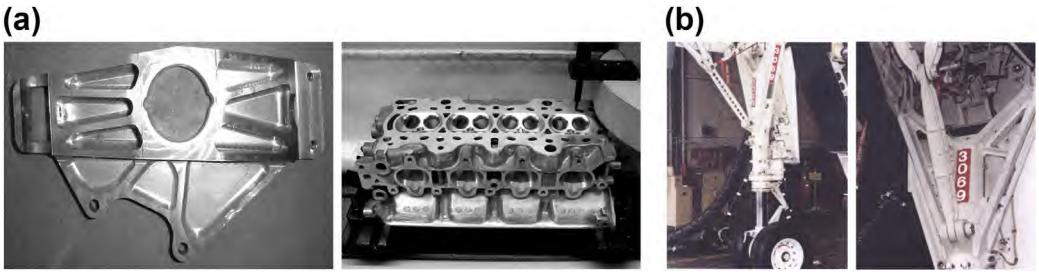


FIGURE 7.1

Structural components of highly complex geometry: (a) automotive suspension component and engine block and (b) airplane landing gear strut.

applicable to structures of arbitrary size and complexity; they employ numerical algorithms for solving differential equations built upon theories of mechanics.

Regardless of approach, the formulation is based on the same three fundamental relations: equilibrium, constitutive, and compatibility. The solutions are approximate when any of these relations is only approximately satisfied or is only an approximation of reality. There always exist uncertainties in modeling and analysis of structural components. It is imperative to account for these uncertainties in order to design more reliable mechanical systems.

In this chapter, we start by briefly reviewing analytical methods and then discuss finite element methods. Essential ingredients in using FEM for modeling and analysis of structural problems, such as mesh generation and boundary conditions, are discussed in detail. We also discuss commercial FEA software, both general-purpose and specialized. Example problems modeled and solved using commercial codes are presented. In this chapter, we also include advanced FEA methods that might be of interest in solving more complex and specialized problems.

7.2 ANALYTICAL METHODS

Analytical methods employ analytical formulations that apply mostly to simple linear elastic problems and lead to closed-form solutions. The solutions are based on linear isotropic infinitesimal elasticity. In other words, they contain assumptions (among others) that the materials in question are elastic, that stress is related linearly to strain, that all deformations are small, and that the material behaves identically regardless of the direction of the applied load (that is, it is *isotropic*). As with any simplifying assumptions in engineering, the more the model strays from reality, the less useful the result. In general, analytical solutions cannot be directly used for support of design. However, they may provide references that support verification of numerical solutions of complex structures.

Three analytical methods commonly employed for structural analysis are briefly discussed. They are often considered in engineering courses such as strength of materials, energy method, and elasticity. We also consider the design criteria of various failure modes, as well as safety factors to address uncertainties and variations.

7.2.1 STRENGTH OF MATERIALS

The strength of materials method is available for simple structural members subject to specific loadings, such as axially loaded bars, prismatic beams in a state of pure bending, bars under direct shear forces, and shafts subject to torsion (Figure 7.2). The solutions can be superimposed using the superposition principle to analyze a member undergoing combined loading following the linear elasticity assumption. Solutions for special cases exist for common structures such as thin-walled pressure vessels.

For the analysis of components and systems other than standard structural members, this method can be used in conjunction with force equilibrium for analytical solutions. An example is the inclined cantilever beam shown in Example 7.1 with its tip constrained to move along the vertical direction. The stress and displacement can be calculated by first decomposing the reaction force at the tip into the axial and transverse directions, and then calculating stress and displacement using standard equations following the superposition principle.

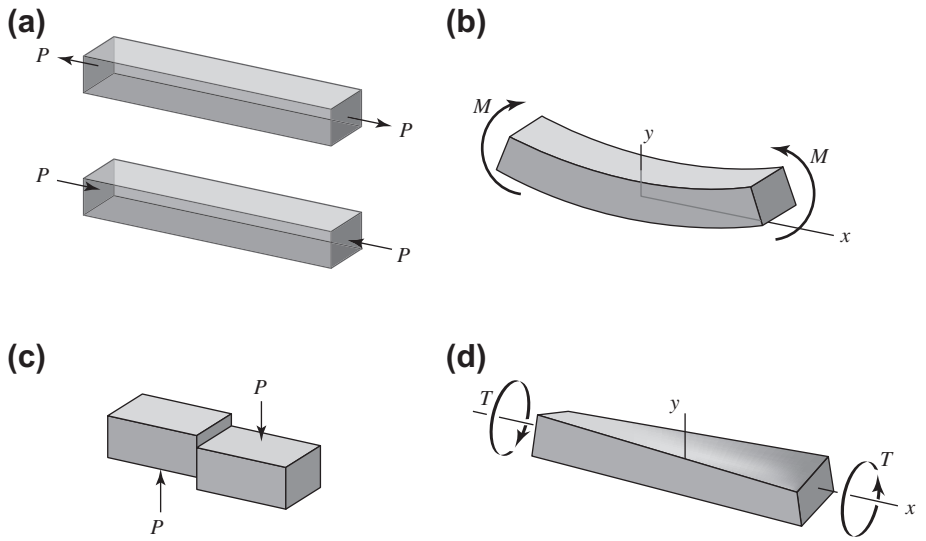
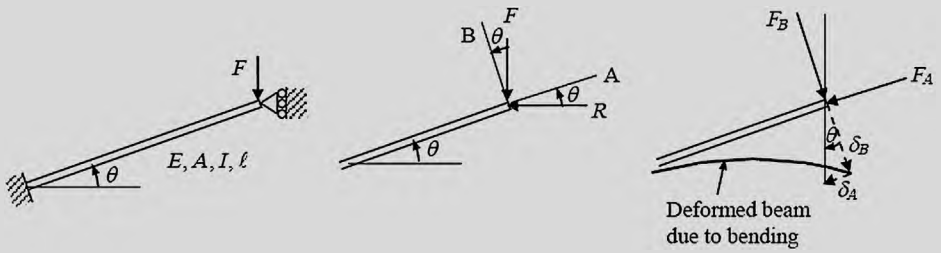


FIGURE 7.2

Basic structural members under loads: (a) axially loaded bars, (b) prismatic beam in a state of pure bending, (c) bar under direct shear, and (d) shaft subject to torsion.

EXAMPLE 7.1

Calculate the displacement at the tip of the inclined cantilever beam shown in the figure below. Note that the tip of the beam is constrained to move vertically. E, A, I, and ℓ are the modulus, cross-sectional area, moment of inertia, and length of the beam, respectively. The cross-section is solid circular with a radius r.



Solutions

This is a statically indeterminate problem. The reaction force R at the tip cannot be solved by force/moment equilibrium equations alone. Instead, it can be obtained by adding the condition imposed by the displacement constraint at the tip. We first draw a free-body diagram for bending force F_B and axial force F_A, and use standard equations for displacements at the tip due to bending and axial forces. We then impose the boundary condition at the tip to solve for the reaction force followed by the displacements.

Continued

EXAMPLE 7.1—cont'd

From force equilibrium, the bending force F_B and axial force F_A can be obtained as

$$F_B = F\cos\theta - R\sin\theta \text{ and } F_A = F\sin\theta + R\cos\theta$$

or simply

$$F_B = Fc - Rs \text{ and } F_A = Fs + Rc$$

From equations provided in Strength of Materials (or Solid Mechanics) textbooks, the displacements due to bending and axial forces are, respectively,

$$\delta_B = \frac{F_B\ell^3}{3EI} = \frac{(Fc - Rs)\ell^3}{3EI} \quad \text{and} \quad \delta_A = \frac{F_A\ell}{EA} = \frac{(Fs + Rc)\ell}{EA}$$

Imposing the displacement constraint at the tip gives the relation between δ_B and δ_A as the following:

$$\frac{\delta_A}{\delta_B} = \tan\theta = \frac{\sin\theta}{\cos\theta} = \frac{s}{c}$$

Thus

$$c\delta_A = s\delta_B \quad \text{or} \quad \frac{c(Fs + Rc)\ell}{EA} = \frac{s(Fc - Rs)\ell^3}{3EI}$$

From the preceding equation, R can be obtained as

$$R = \frac{Fsc(A\ell^2 - 3I)}{Q}$$

where $Q = 3Ic^2 + A\ell^2s^2$, and

$$\delta_B = \frac{(Fc - Rs)\ell^3}{3EI} = \frac{Fc\ell^3}{EQ} \quad \text{and} \quad \delta_A = \frac{Fs\ell^3}{EQ}$$

Therefore, the vertical displacement at the tip is

$$\delta = \frac{\delta_B}{c} = \frac{F\ell^3}{EQ} = \frac{F\ell^3}{E(3Ic^2 + A\ell^2s^2)}$$

Note that stress in the beam can be calculated by superposing bending and axial stresses due to the respective bending force F_B and axial force F_A. The stress calculation is left as an exercise (Exercise 7.1).

7.2.2 ENERGY METHOD

Energy principles in structural mechanics express the relationships between stresses, strains or deformations, displacements, material properties, and external effects in the form of energy or work done by internal and external forces. Since energy is a scalar quantity, these relationships provide convenient and alternative means for formulating the governing equations of deformable bodies in solid mechanics. The energy method is very general and is applicable to many structural problems, especially for dynamic responses and nonlinear analysis. One of its simplest forms is Castigliano's theorem, which is powerful for solving more complex elastostatic problems.

Strain energy must first be calculated before applying Castigliano's theorem. The strain energy for standard structural members is given in Figure 7.3. For a structure component loaded with a force P or torque/moment M, the theorem states:

> The partial derivative of the strain energy, considered as a function of the applied forces (and moments) acting on a linear elastic structure, with respect to one of these forces (or moments), is equal to the displacement (or rotation angle) in the direction of the force (or moment) of its point of application.

Mathematically, the theorem states:

$$\delta = \frac{\partial U}{\partial P} \quad \text{or} \quad \theta = \frac{\partial U}{\partial M} \tag{7.1}$$

where

δ is the displacement (or θ the rotation angle) in the direction of force P (or moment M) of its point of application
U is the strain energy.

Note that for a built-up structure (consisting of multiple components) U is the sum of the strain energy of its constituent components.

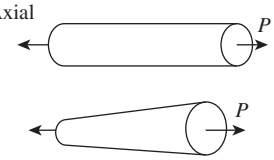
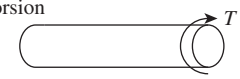
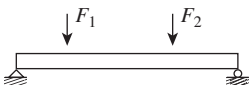
(a)	Prismatic bar with constant force Nonprismatic bar (general)	Axial 	$U = \frac{L}{2EA} P^2$ $U = \int_0^L \frac{P^2}{2EA} dx^2$
(b)	Beam bending (general)	Moment, vertical force 	$U = \int_0^L \frac{M^2}{2EI} dx$
(c)	Torsional shear (circular)	Torsion 	$U = \frac{L}{2GJ} T^2$
(d)	Transverse shear (general)	Moment, vertical force 	$U = \int_0^L \frac{CV^2}{2AG} dx$ C = 1.20 (rectangular), 1.11 (circular), 2.00 (thin wall, round)

FIGURE 7.3

Strain energy of basic structural members: (a) truss of axial load; (b) beam of pure bending; (c) shaft of pure torsion; and (d) transverse shear.

Example 7.2 illustrates how the energy theory can be applied to solve a structural problem.

EXAMPLE 7.2

Calculate the displacement at the tip of the inclined cantilever beam discussed in Example 7.1 using Castigliano's theorem.

Solutions

The tip displacement can be obtained using $\delta = \frac{\partial U}{\partial F}$, where U is the strain energy of the beam due to the applied force F and the reaction force R. Note that the beam bends and shrinks as a result of these two forces, respectively; the strain energy U can be found using (see Figure 7.3)

$$U = \frac{\ell}{2EA}P^2 + \int_0^\ell \frac{M^2}{2EI}dx$$

In this case, the axial force P is $F_A = Fs + Rc$, and the bending moment M is due to the transverse force F_B. Hence the strain energy is

$$U = \frac{\ell}{2EA}F_A^2 + \int_0^\ell \frac{(F_B x)^2}{2EI}dx = \frac{\ell}{2EA}(Fs + Rc)^2 + \int_0^\ell \frac{[(Fc - Rs)x]^2}{2EI}dx$$

$$= \frac{\ell}{2EA}(Fs + Rc)^2 + \frac{(Fc - Rs)^2\ell^3}{6EI}$$

and the displacement at the tip, according to Castigliano's theorem is

$$\delta = \frac{\partial U}{\partial F} = \frac{\ell}{EA}(Fs + Rc)\left(s + \frac{\partial R}{\partial F}c\right) + \frac{\ell^3}{3EI}(Fc - Rs)\left(c - \frac{\partial R}{\partial F}s\right) = \frac{F\ell^3}{EQ}$$

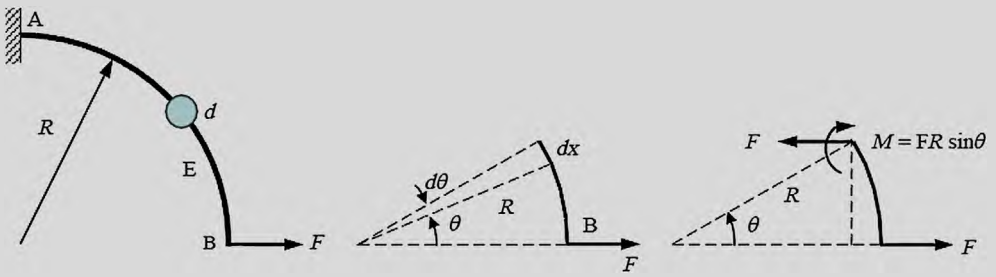
where

$$\frac{\partial R}{\partial F} = \frac{sc(A\ell^2 - 3I)}{Q}$$

As can be seen in Example 7.2, the formulation of the energy method for structural analysis is more general and concise. Even though the algebra involved may be about the same as in the strength of materials approach, the energy method is more powerful and is able to solve complex problems—for example, a quarter-arc beam under a point force as shown in Example 7.3.

EXAMPLE 7.3

Calculate the horizontal displacement δ_B at the tip of the cantilever beam of 90-degree circular arc shown in the figure below. We assume that the diameter of the cross-section d is small compared with the radius R of the beam, so we may ignore the transverse shear effect.



EXAMPLE 7.3—cont'd

Solutions

The strain energy for the curved beam can be easily converted from the Cartesian coordinate system to the polar coordinate system as

$$U = \int_0^{\ell} \frac{M^2}{2EI} dx = \int_0^{\theta} \frac{M^2}{2EI} R d\theta$$

where $dx = Rd\theta$. Furthermore, from the free-body diagram, the internal moment of the beam is $M = FR\sin\theta$. Therefore, the horizontal displacement at the tip B is

$$h_B = \frac{\partial U}{\partial F} = \frac{\partial}{\partial F} \int_0^{\pi/2} \frac{(FR\sin\theta)^2}{2EI} R d\theta = \frac{FR^3}{EI} \int_0^{\pi/2} (\sin\theta)^2 d\theta$$

$$= \frac{FR^3}{EI} \int_0^{\pi/2} \frac{1}{2}\left(1 - \cos 2\theta\right) d\theta = \frac{FR^3}{2EI}\left(\theta - \frac{1}{2}\sin 2\theta\right)_0^{\pi/2}$$

$$= \frac{\pi FR^3}{4EI}$$

Note that it would be difficult to solve this problem using the basic strength of materials approach.

7.2.3 LINEAR ELASTICITY

Elasticity is the mathematical study of how solid objects deform and become internally stressed as a result of prescribed loading conditions. Elasticity relies on the continuum hypothesis and is applicable at macroscopic length scales. Linear elasticity is a simplification of the more general nonlinear theory of elasticity and is a branch of continuum mechanics. The fundamental "linearizing" assumptions of linear elasticity are: infinitesimal strains or "small" deformations (or strains) and the linear relationships between the components of stress and strain.

The goal of solving a problem in elasticity is usually to find the stress distribution in an elastic body and, in some cases, to find the strain at any point due to given body forces and prescribed conditions at the body's boundary. To determine the stress at a point in, for example, a 2D body (Figure 7.4), we

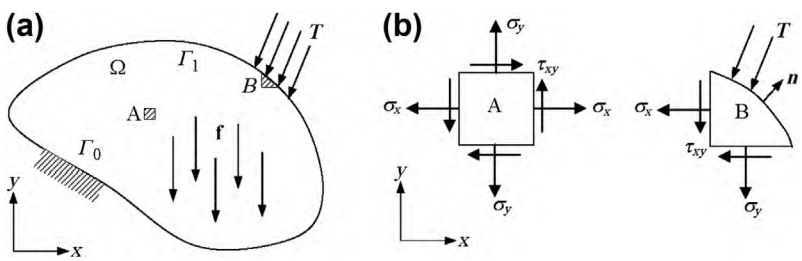


FIGURE 7.4

A loaded 2D structure: (a) load and boundary conditions and (b) stress elements A and B.

must find three stress components, σ_x, σ_y, and τ_{xy}. These components satisfy two equilibrium equations:

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + f_x = 0 \quad \text{and} \quad \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + f_y = 0 \tag{7.2a}$$

with boundary conditions

$$\tau_{xy} n_y + T_x = 0 \quad \text{and} \quad \tau_{xy} n_x + T_y = 0 \text{ on } \Gamma_1 \tag{7.2b}$$

$$u_x = u_y = 0 \text{ on } \Gamma_0 \tag{7.2c}$$

where f_x and f_y are the body forces (or gravitational forces), $\boldsymbol{n}$ is the surface normal vector at the traction boundary (Γ_1), and $\boldsymbol{T} = [T_x, T_y]^T$ is the traction force. Note that at boundary Γ_0, the displacements are $u_x = u_y = 0$.

Since two equations (Eq. 7.2a) are not sufficient to solve for the three unknowns (σ_x, σ_y, and τ_{xy}), we introduce the three strain components ε_x, ε_y, and γ_{xy}. At the same time we have the three relations defining the strain components in terms of the two displacement components u and v in the x- and y-directions, respectively:

$$\varepsilon_x = \frac{\partial u}{\partial x}, \quad \varepsilon_y = \frac{\partial v}{\partial y}, \quad \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \tag{7.3}$$

Equation 7.3 presents three equations for only two unknowns (u and v); we cannot expect that in general that these equations will have a solution if the strain components are arbitrarily prescribed. The strains must satisfy the so-called compatibility condition, implying that the underlying displacement functions governed by the strains ensure a continuously deformed body. For a 2D planar structure, the compatibility equation is

$$\frac{\partial^2 \varepsilon_x}{\partial y^2} + \frac{\partial^2 \varepsilon_y}{\partial x^2} = \frac{\partial^2 \gamma_{xy}}{\partial x \partial y} \tag{7.4}$$

In addition, we have three stress-strain relations, for example, for a plane stress problem:

$$\varepsilon_x = \frac{1}{E}(\sigma_x - \nu \sigma_y), \quad \varepsilon_y = \frac{1}{E}(\sigma_y - \nu \sigma_x), \quad \gamma_{xy} = \frac{\tau_{xy}}{G} \tag{7.5}$$

Thus we have altogether eight unknowns (σ_x, σ_y, τ_{xy}, u, v, ε_x, ε_y, and γ_{xy}) and eight equations, combining Eqs 7.2 through 7.5. This system of equations is generally sufficient for the solution of an elasticity problem.

Formulation of the elasticity problem is very general regardless of the geometry of the structure. However, solving this system of equations is not straightforward. One of the possibilities is to introduce a stress function $\psi(x,y)$ and focus only on solving the three stress components. The stress function must satisfy Eq. 7.2, such that

$$\sigma_x = \frac{\partial^2 \psi}{\partial y^2}, \quad \sigma_y = \frac{\partial^2 \psi}{\partial x^2}, \quad \tau_{xy} = -\frac{\partial^2 \psi}{\partial x \partial y} \tag{7.6}$$

neglecting the gravitational force $\boldsymbol{f}$.

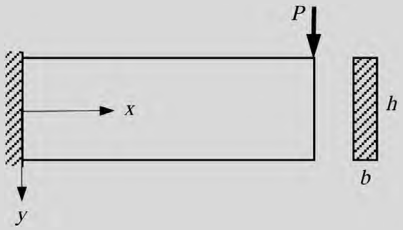
Also, the stress function must represent the strain components that satisfy the compatibility equation. If we insert Eq. 7.6 into Eq. 7.5 and then into Eq. 7.4, we have

$$\frac{\partial^4 \psi}{\partial x^4} + 2\frac{\partial^4 \psi}{\partial x^2 \partial y^2} + \frac{\partial^4 \psi}{\partial y^4} = \nabla^4 \psi = 0 \tag{7.7}$$

Our problem thus reduces to the determination of the stress function ψ with appropriate boundary conditions. Once the stress function is found, the stress components can be determined by Eq. 7.6. Example 7.4 illustrates how the elasticity method can be applied to solve structural problems.

EXAMPLE 7.4

A cantilever beam of narrow rectangular cross-section under an end load P is shown in the figure below. With its width b small compared with its depth h, the loaded beam may be regarded as an example of plane stress. The boundary conditions are that the upper and lower edges are free from load and that the resulting shear force at $x = 0$ is equal to P. Find the three stress components, σ_x, σ_y, and τ_{xy}.



Solutions

The key to solving this problem using the elasticity formulation is selecting an adequate stress function ψ. Since the bending stress σ_x is linear in terms of both x and y, it is reasonable to assume a trial function ψ as

$$\sigma_x = \frac{\partial^2 \psi}{\partial y^2} = c_1 xy$$

where c_1 is a constant to be determined using boundary conditions. Integrating twice over y yields

$$\psi = \frac{c_1}{6} xy^3 + yf_1(x) + f_2(x)$$

where f_1 and f_2 are unknown functions of x. Substitution of this expression into Eq. 7.7 yields

$$y\frac{d^4 f_1}{dx^4} + \frac{d^4 f_2}{dx^4} = 0$$

Since f_1 and f_2 are functions of x alone, the second term here is independent of y. This is possible only if

$$\frac{d^4 f_1}{dx^4} = 0 \quad \text{and} \quad \frac{d^4 f_2}{dx^4} = 0$$

or if $f_1 = c_2x^3 + c_3x^2 + c_4x + c_5$ and $f_2 = c_6x^3 + c_7x^2 + c_8x + c_9$, where c_2 through c_9 are constants of integration. Therefore, the stress function ψ is

$$\psi = \frac{c_1}{6} xy^3 + y(c_2x^3 + c_3x^2 + c_4x + c_5) + (c_6x^3 + c_7x^2 + c_8x + c_9)$$

From Eq. 7.6,

$$\sigma_y = \frac{\partial^2 \psi}{\partial x^2} = 6(c_2y + c_6)x + 2(c_3y + c_7)$$

Continued

EXAMPLE 7.4—cont'd

$$\tau_{xy} = -\frac{\partial^2 \psi}{\partial x \partial y} = -\frac{c_1}{2}y^2 - 3c_2x^2 - 2c_3x - c_4$$

The boundary conditions require that $\sigma_y = 0$ at $y = \pm h/2$, yielding $c_2 = c_3 = c_6 = c_7 = 0$. Thus,

$$\tau_{xy} = -\frac{c_1}{2}y^2 - c_4$$

Again, imposing boundary conditions for $\tau_{xy} = 0$ at $y = \pm h/2$ yields $c_4 = -c_1 h^2/8$. Also, at the loaded end of the beam the sum of the shear force must be equal to P:

$$\int_{-h/2}^{h/2} \tau_{xy} b dy = \int_{-h/2}^{h/2} \frac{c_1}{8} b(4y^2 - h^2) dy = P$$

from which $c_1 = \frac{-12P}{bh^3}$. Note that $I = bh^3/12$ is the moment of inertia of the rectangular cross-section.

The final expressions for the stress components are therefore

$$\sigma_x = -\frac{Pxy}{I}, \sigma_y = 0 \quad \text{and} \quad \tau_{xy} = -\frac{P}{2I}\left(\frac{h^2}{4} - y^2\right)$$

This coincides with the solutions given by the strength of materials approach.

Once the stress components are solved, the strain components can be obtained from Eq. 7.5 and then the displacement functions from Eq. 7.3. Note that, in solving displacements, integration constants must be determined by displacement conditions. This is left as an exercise (Exercise 7.3).

Question: What if an incorrect stress function were selected? What would happen if

$$\sigma_x = \frac{\partial^2 \psi}{\partial y^2} = c_1 x^2 y$$

or

$$\sigma_x = \frac{\partial^2 \psi}{\partial y^2} = c_1 x^2$$

were chosen? Again, this is left as an exercise (Exercise 7.4).

7.2.4 FAILURE CRITERIA

It is well known that failure of a tensile member occurs when the stress caused by the actual load reaches the stress limit—that is, the strength of the member's material (usually yield strength S_y for ductile materials and ultimate tensile strength S_u for brittle materials). Correlation of the actual stress with the material strength, which is the maximum stress that the structural member is able to bear without a failure, is straightforward in this case because they are both uniaxial. But how can we correlate the biaxial or triaxial stress state in a component—whose material strength is measured in

uniaxial tests—to assess failure tendency? This is where failure theories come into the picture. One of the most common theories in design is the maximum shear stress theory, which states:

> Failure will occur in a complex part if any of the maximum shear stresses τ_{max} exceeds the material shear yield strength S_{sy} that causes failure in the simple, uniaxial test.

That is, the maximum shear stress must be smaller than the shear yield strength to ensure a safe design.

Mathematically, the theorem says:

$$\tau_{\max} \leq S_{sy} \tag{7.8}$$

What is S_{sy}? For the uniaxial stress case shown in Figure 7.5(a), when the stress of a stress element exceeds the stress limit—for example, the yield strength for ductile materials S_y—the structure fails. If we rotate the stress element 45° clockwise (i.e., equivalently rotate the stress point 90° clockwise from X to X' on Mohr's circle) as shown in Figure 7.5(b), the shear stress τ_{max} becomes

$$\tau_{\max} = \frac{\sigma_1}{2} \leq \frac{S_y}{2}$$

Therefore, referring to Eq. 7.8, we have $S_{sy} = S_y/2$. Also, from strength of materials, the maximum stress for a biaxial stress element is

$$\tau_{\max} = \frac{1}{2}|\sigma_1 - \sigma_2| \tag{7.9}$$

where σ_1 and σ_2 are the two principal stresses.

Graphically, the maximum shear stress theory for a biaxial stress state can be depicted in a Tresca's hexagon (Figure 7.6), which states a safe zone as

$$|\sigma_1| \quad \text{and} \quad |\sigma_2| \leq S_y \tag{7.10a}$$

for stress points (σ_1, σ_2) falling in the first and third quadrants, and

$$\tau_{\max} = \frac{1}{2}|\sigma_1 - \sigma_2| \leq \frac{S_y}{2} \tag{7.10b}$$

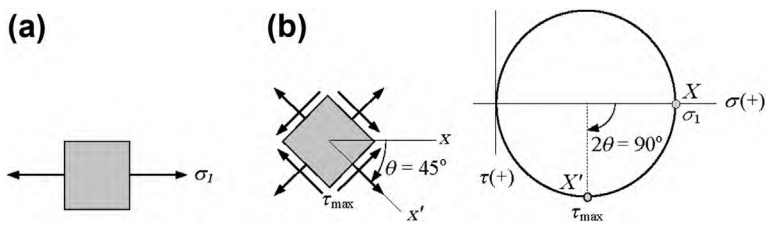


FIGURE 7.5

(a) Stress element under uniaxial load and (b) rotated stress element with Mohr's circle.

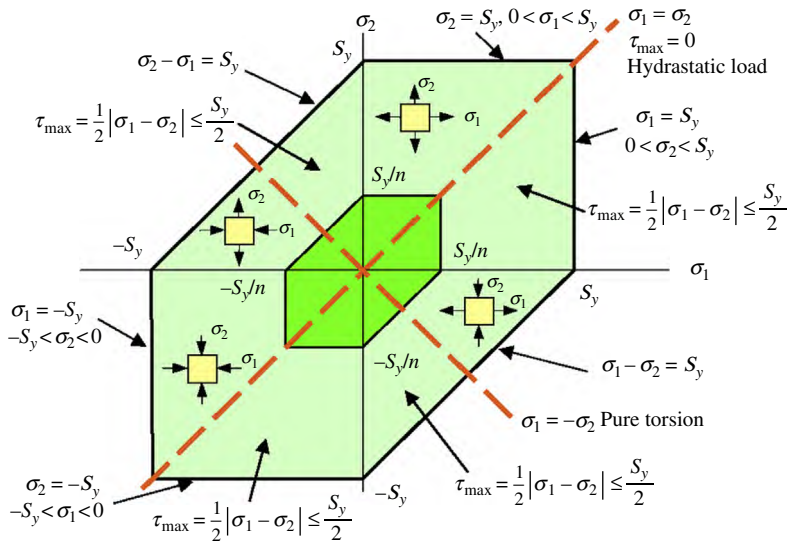


FIGURE 7.6

Tresca's hexagon.

for those in the second and fourth quadrants. That the principal stresses fall inside the hexagon indicates a safe design. The smaller hexagon indicates a safe zone when a larger safety factor n is considered.

Note that the maximum shear stress theory is applicable to ductile materials. Others are applicable to brittle materials. Figure 7.7 provides useful and popular theories, as well as how criteria should be selected. As a rule of thumb, ductile and brittle materials are separated by strain at fracture. For ductile materials, the strain is greater than 5% at fracture. Usually brittle materials fracture with a strain less than 5%. More details about the failure criteria shown in Figure 7.7 can be found in either a strength of materials textbook, such as Beer et al. (2002), or a design of mechanical components textbook—for example Hamrock et al. (2002).

7.2.5 UNCERTAINTIES, VARIATIONS, AND SAFETY FACTORS

Whether analytical or numerical, structural analysis assumes no variations in physical parameters and no uncertainties in physical conditions. In reality, loads, material properties, and geometric dimensions vary, and there are uncertainties in determining boundary conditions, making assumptions in converting physical problems to a mathematically solvable form, and employing analysis methods for problem solution. It is now widely recognized that a quantitative assessment of the effects of uncertainties in structural analysis and design plays an important role in quality assurance and reliability estimation. Variations and uncertainties are often addressed using more conservative approaches, such as safety factor and worst-case scenario. However, these approaches could lead to overdesign. In this section we briefly discuss the safety factor approach. Probabilistic analysis and reliability estimate will be discussed in Chapter 10 Reliability Analysis.

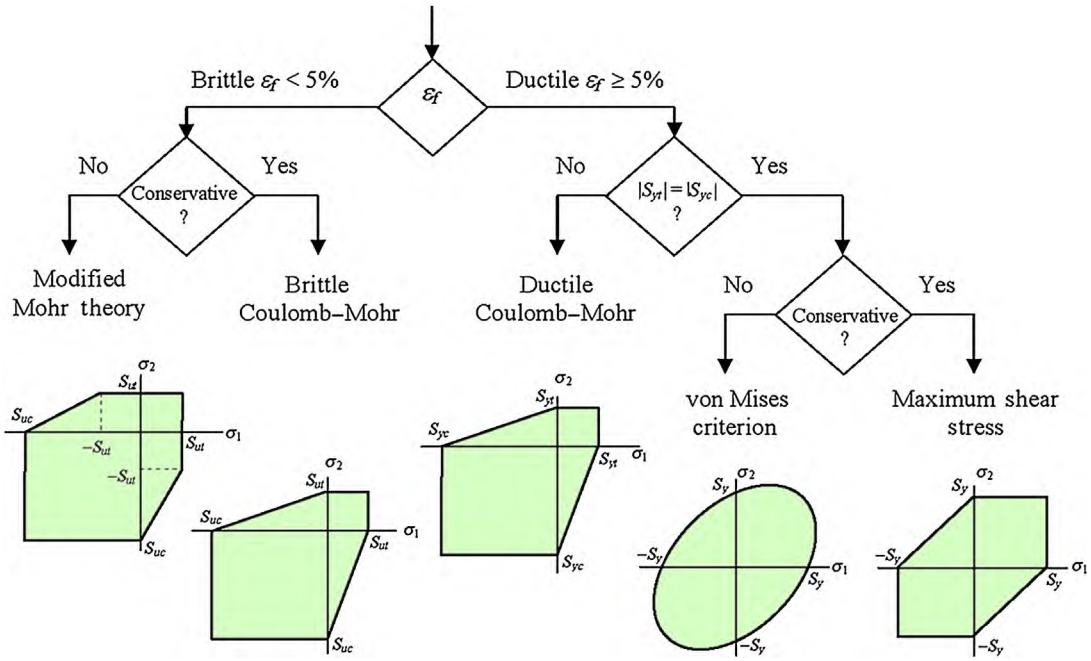


FIGURE 7.7

Flowchart for choosing proper failure criteria for design.

Variability covers the variation inherent to the modeled physical system or the environment under consideration. Generally, this is described by a distributed quantity defined over a range of possible values. The exact value is known to be within this range, but it varies from unit to unit or from time to time. Ideally, objective information on both the range and the likelihood of the quantity within this range is available. This type of nondeterminism is also referred to as aleatory, irreducible, stochastic, or due to chance.

Uncertainty is a potential deficiency in any phase or activity of the modeling process that derives from lack of knowledge. The word *potential* emphasizes that the deficiency may or may not occur. This definition basically states that uncertainty is caused by incomplete information resulting from vagueness, nonspecificity, or dissonance. Vagueness characterizes information that is imprecisely defined, unclear, or indistinct. It is typically the result of human opinion regarding unknown quantities. In the literature, this type of nondeterminism is also referred to as epistemic, reducible, or subjective.

On a relative scale, variability is easier to quantify. Recently developed reliability analysis methods have been largely employed to incorporate stochastic or variability of physical parameters in failure probability estimates for product design. However, quantifying and addressing uncertainty in engineering design, especially that caused by incomplete information, is not as straightforward.

The simplest and most widely used approach in addressing variability and uncertainty in engineering design is referred to as factor of safety. In employing this approach designers often overdesign the product in hope of overcoming potential problems caused by variability and uncertainty.

The sources of variability and uncertainty are classified into geometric simplification, material modeling, level of sophistication of analysis, and human error. Determining an adequate safety factor for product design often requires numerous considerations, such as the degree of uncertainty about loading, the degree of uncertainty about material strength, the uncertainties in relating applied loads to material strength via stress analysis, the consequences of failure (e.g., human safety and economics), and the cost of providing a large safety factor (Norton, 2011). Determining safety factors for engineering design is not completely science. Industry has developed certain guidelines for safety factors based on the past experience and failure rate. The following are general guidelines for engineering students (Juvinall and Marshek, 2000).

- $n = 1 \sim 2$ for reliable materials used under stable conditions subject to reliable loads (hence variability and uncertainly in stress).
- $n = 2 \sim 3$ for less tried materials used under average conditions, subject to fairly reliable loads.
- $n = 3 \sim 4$ for at most one uncertainty in materials, conditions, and loads.
- $n = 1 \sim 6$ for fatigue (applied to endurance limit).
- $n = 3 \sim 6$ for impact.

7.3 FINITE ELEMENT METHODS

The finite element method, or analysis, (FEM or FEA) is a numerical technique for finding approximate solutions to partial differential equations (PDEs) or sometimes integral equations that govern a physical problem.

The term *finite element* was first coined by Clough in 1960. In the early 1960s, engineers used the method for approximate solutions of problems in stress analysis, fluid flow, heat transfer, and other areas. The first book on FEM, by Zienkiewicz and Chung, was published in 1967. In the late 1960s and early 1970s, FEM was applied to a wide variety of engineering problems.

The finite element method originated from the need for solving complex elasticity and structural analysis problems in civil and aeronautical engineering. Its development began in the middle to late 1950s for airframe and structural analysis. By the late 1950s, the key concepts of stiffness matrix and element assembly existed essentially in the form used today. NASA issued a request for proposals for the development of the finite element software Nastran in 1965. The method was provided with a rigorous mathematical foundation in 1973 (Strang and Fix, 1973) and has since been generalized into a branch of applied mathematics for numerical modeling of physical systems in a wide variety of engineering disciplines, such as electromagnetism and fluid dynamics. Recently the method was further extended in various mathematical forms, such as p-version FEM (Szabó and Babuška, 1991), the meshless method (Li and Liu, 2004, Belytschko and Chen, 2007), and extended FEM (Moës et al., 1999).

Although FEM has been generalized and applied to solve various engineering problems, in this chapter we focus on structural problems. We start with a simple example to illustrate the basic concept of FEA, and then discuss general formulations for both its h- and p-versions. The important concept of solution convergence, in the h- and p-methods, as well as adaptations, is also treated.

7.3.1 A SIMPLE EXAMPLE

The formulation of the finite element method starts with the principle of virtual work. Virtual work is the work done by real forces in moving through virtual displacements. A virtual displacement is any displacement consistent with the constraints of the structure—that is, it satisfies the boundary conditions at the supports, or the *essential boundary conditions*. The principle of virtual work states that if a solid is in equilibrium then the virtual work done in any virtual displacement that satisfies the essential boundary conditions is zero.

For the cantilever beam example shown in Figure 7.8, the governing differential equation is known from strength of materials:

$$\frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2 w}{\partial x^2}\right) = q + P\widehat{\delta}(\ell - x) \tag{7.11a}$$

where $\widehat{\delta}(\ell - x)$ is the Dirac delta function, and $\int_{-\infty}^{\infty} \widehat{\delta}(\ell - x)f(x)dx = f(\ell)$. The boundary conditions are

$$w(0) = 0 \quad \text{and} \quad \frac{\partial w}{\partial x}(0) = 0 \tag{7.11b}$$

$$EI\frac{\partial^2 w}{\partial x^2}(\ell) = 0 \quad \text{and} \quad \frac{\partial}{\partial x}\left(EI\frac{\partial^2 w}{\partial x^2}\right)_{x=\ell} = 0 \tag{7.11c}$$

Note that Eq. 7.11b shows the essential boundary conditions that the virtual displacement must satisfy. Certainly, the true solutions must satisfy both conditions stated in Eqs 7.11b and 7.11c.

Multiply the virtual displacement $\delta w(x)$ on both sides of Eq. 7.11a, and integrate over the beam length. On the right side, we have

$$\int_0^{\ell} (q + P\widehat{\delta}(\ell - x))\delta w dx = \int_0^{\ell} q\delta w dx + \int_0^{l} P\widehat{\delta}(\ell - x)\delta w(x)dx = \int_0^{l} q\delta w dx + P\delta w(\ell) \tag{7.12}$$

This is the virtual work done by the externally applied forces in moving through the virtual displacement $\delta w(x)$.

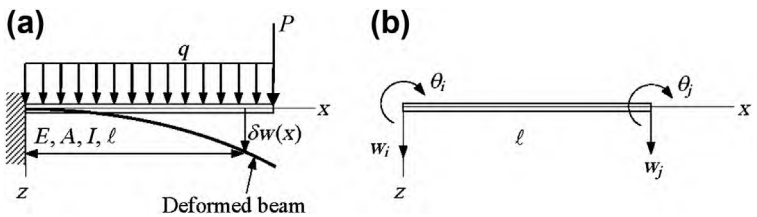


FIGURE 7.8

Simple cantilever beam: (a) load and virtual displacement and (b) beam element with displacement and rotation degrees of freedom.

Twice integrating the term on the left side by parts yields

$$\int_0^l \left(\frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2 w}{\partial x^2}\right)\right)\delta w dx = \frac{\partial}{\partial x}\left(EI\frac{\partial^2 w}{\partial x^2}\right)\delta w\Big|_0^\ell - \int_0^\ell \left(\frac{\partial}{\partial x}\left(EI\frac{\partial^2 w}{\partial x^2}\right)\right)\frac{\partial \delta w}{\partial x}dx$$
$$= \frac{\partial}{\partial x}\left(EI\frac{\partial^2 w}{\partial x^2}\right)\delta w\Big|_0^\ell - EI\frac{\partial^2 w}{\partial x^2}\frac{\partial \delta w}{\partial x}\Big|_0^\ell + \int_0^\ell EI\frac{\partial^2 w}{\partial x^2}\frac{\partial^2 \delta w}{\partial x^2}dx \tag{7.13}$$

The first two terms at $x = \ell$ vanish because of the boundary conditions of Eq. 7.11c that require that the true solutions comply; these are the so-called natural boundary conditions. If the virtual displacement δw belongs to a *kinematically admissible virtual displacement space* Z, defined as $Z = \{u \in C^1, u = 0 \text{ and } \partial u/\partial x = 0 \text{ at } x = 0\}$, the first two terms at $x = 0$ also vanish. Note that C^1 is a space of functions with first-order derivative continuous. The remaining term is nothing but the work done by the internal forces.

Therefore, the equilibrium equation, Eq. 7.11a, can be written as

$$\int_0^\ell EI\frac{\partial^2 w}{\partial x^2}\frac{\partial^2 \delta w}{\partial x^2}dx = \int_0^\ell q\delta w dx + P\delta w(\ell), \quad \text{for all } \delta w \in Z. \tag{7.14}$$

These generalized formulations are often referred to as *weak*, which is essentially what the principle of virtual work states.

To solve Eq. 7.14, we convert the integral equation into a set of linear equations in matrix form by introducing interpolation functions (or shape functions) to represent the displacement fields in the finite elements.

For a beam element under bending, shown in Figure 7.8(b), there are two degrees of freedom at each end point, or *node*: the displacement w and the rotation angle θ. From strength of materials, displacement of a cantilever beam with a point load at the tip is a cubic function in the length parameter. Therefore, in FEA a cubic shape function is employed for a beam element. For a cantilever beam with a point load at the tip, one beam element is sufficient to provide an exact solution. However, for a beam with a uniformly distributed load, the exact displacement solution is a fourth-order function in beam length. One element of the cubic shape function does not give an exact solution. In fact, for the example shown in Figure 7.8, where a distributed load is present, exact solutions are obtained at nodes but not in-between.

Using a cubic shape function, the displacement function $w(x)$ in the beam element can be interpolated as

$$w(x) = \boldsymbol{N}^{\mathrm{T}}\boldsymbol{W} = [N_1 N_2 N_3 N_4]\begin{bmatrix} w_i \\ \theta_i \\ w_j \\ \theta_j \end{bmatrix} \tag{7.15}$$

where

$$N_1 = 1 - 3\frac{x^2}{\ell^2} + 2\frac{x^3}{\ell^3}, \quad N_2 = x - 2\frac{x^2}{\ell} + \frac{x^3}{\ell^2},$$
$$N_3 = 3\frac{x^2}{\ell^2} - 2\frac{x^3}{\ell^3}, \quad N_4 = -\frac{x^2}{\ell} + \frac{x^3}{\ell^2} \tag{7.16}$$

and

$$\frac{\partial^2 w}{\partial x^2} = \frac{\partial^2 \boldsymbol{N}^{\mathrm{T}}}{\partial x^2}\boldsymbol{W} = \left[\frac{-6}{\ell^2} + 12\frac{x}{\ell^3}\ \frac{-4}{\ell} + 6\frac{x}{\ell^2}\ \frac{6}{\ell^2} - 12\frac{x}{\ell^3}\ \frac{-2}{\ell} + 6\frac{x}{\ell^2}\right]\begin{bmatrix} w_i \\ \theta_i \\ w_j \\ \theta_j \end{bmatrix}$$

Similarly, $\frac{\partial^2 \delta w}{\partial x^2} = \frac{\partial^2 \boldsymbol{N}^{\mathrm{T}}}{\partial x^2}\delta\boldsymbol{W}$. Inserting these into Eq. 7.14 and carrying out the integrations, we have

$$\int_0^\ell EI\frac{\partial^2 w}{\partial x^2}\frac{\partial^2 \delta w}{\partial x^2}dx = \delta\boldsymbol{W}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{W}$$

where $\boldsymbol{K}$ is the so-called *stiffness matrix*; that is,

$$\boldsymbol{K} = \frac{EI}{\ell^3}\begin{bmatrix} 12 & 6\ell & -12 & 6\ell \\ 6\ell & 4\ell^2 & -6\ell & 2\ell^2 \\ -12 & -6\ell & 12 & -6\ell \\ 6\ell & 2\ell^2 & -6\ell & 4\ell^2 \end{bmatrix}$$

and

$$\int_0^\ell q\delta w dx + P\delta w(\ell) = \delta\boldsymbol{W}^{\mathrm{T}}\int_0^\ell q\boldsymbol{N}dx + \delta\boldsymbol{W}^{\mathrm{T}}\begin{bmatrix} 0 \\ 0 \\ P \\ 0 \end{bmatrix} = \delta\boldsymbol{W}^{\mathrm{T}}\begin{bmatrix} \frac{1}{2}q\ell \\ \frac{1}{12}q\ell^2 \\ \frac{1}{2}q\ell + P \\ -\frac{1}{12}q\ell^2 \end{bmatrix} = \delta\boldsymbol{W}^{\mathrm{T}}\boldsymbol{F} \tag{7.18}$$

where $\boldsymbol{F}$ is the *force vector*. Equation 7.14 becomes $\delta\boldsymbol{W}^{\mathrm{T}}\boldsymbol{K}\boldsymbol{W} = \delta\boldsymbol{W}^{\mathrm{T}}\boldsymbol{F}$. Since $\delta\boldsymbol{W}$ is a virtual displacement, we have

$$\boldsymbol{K}\boldsymbol{Z} = \boldsymbol{F}$$

or

$$\frac{EI}{\ell^3}\begin{bmatrix} 12 & 6\ell & -12 & 6\ell \\ 6\ell & 4\ell^2 & -6\ell & 2\ell^2 \\ -12 & -6\ell & 12 & -6\ell \\ 6\ell & 2\ell^2 & -6\ell & 4\ell^2 \end{bmatrix}\begin{bmatrix} w_i \\ \theta_i \\ w_j \\ \theta_j \end{bmatrix} = \begin{bmatrix} \frac{1}{2}q\ell \\ \frac{1}{12}q\ell^2 \\ \frac{1}{2}q\ell + P \\ -\frac{1}{12}q\ell^2 \end{bmatrix} \tag{7.19}$$

If we impose the essential boundary conditions—in this case, $w_i = \theta_i = 0$—the reduced matrix equations become

$$\frac{EI}{\ell^3}\begin{bmatrix} 12 & -6\ell \\ -6\ell & 4\ell^2 \end{bmatrix}\begin{bmatrix} w_j \\ \theta_j \end{bmatrix} = \begin{bmatrix} \frac{1}{2}q\ell + P \\ -\frac{1}{12}q\ell^2 \end{bmatrix}$$

Invert the matrix, and solve the equations:

$$\begin{bmatrix} w_j \\ \theta_j \end{bmatrix} = \frac{\ell^3}{EI}\begin{bmatrix} 12 & -6\ell \\ -6\ell & 4\ell^2 \end{bmatrix}^{-1}\begin{bmatrix} \frac{1}{2}q\ell + P \\ -\frac{1}{12}q\ell^2 \end{bmatrix} = \frac{\ell}{12EI}\begin{bmatrix} 4\ell^2 & 6\ell \\ 6\ell & 12 \end{bmatrix}\begin{bmatrix} \frac{1}{2}q\ell + P \\ -\frac{1}{12}q\ell^2 \end{bmatrix} = \begin{bmatrix} \frac{q\ell^4}{8EI} + \frac{P\ell^3}{3EI} \\ \frac{q\ell^3}{6EI} + \frac{P\ell^2}{2EI} \end{bmatrix} \tag{7.20}$$

This is identical to the exact solutions at the tip while combining the tip displacements of the separate loading cases: point load P and distributed load q.

The finite element solution for displacement $w(x)$ can then be obtained from Eq. 7.15:

$$w(x) = \boldsymbol{N}^{\mathrm{T}}\boldsymbol{W} = [N_3 \;\; N_4]\begin{bmatrix} w_j \\ \theta_j \end{bmatrix} = \left[3\frac{x^2}{\ell^2} - 2\frac{x^3}{\ell^3}, -\frac{x^2}{\ell} + \frac{x^3}{\ell^2}\right]\begin{bmatrix} \frac{q\ell^4}{8EI} + \frac{P\ell^3}{3EI} \\ \frac{q\ell^3}{6EI} + \frac{P\ell^2}{2EI} \end{bmatrix}$$

$$= \frac{q\ell}{24EI}(5x^2\ell - 2x^3) + \frac{P}{6EI}(3x^2\ell - x^3)$$

where the first and second terms represent the beam displacements due to the distributed load q and the point force P, respectively. From strength of materials, the second term is exact, but the exact solution for the distributed load q is

$$w^q(x) = \frac{q}{24EI}(6x^2\ell^2 - 4x^3\ell + x^4)$$

Apparently, the finite element method does not give an exact solution for $w^q(x)$ because the element shape function is a cubic function and the exact solution is a fourth-order polynomial function. However, when more elements are employed for the cantilever beam (i.e., by dividing the beam into smaller segments), the finite element solution approaches the exact solution.

7.3.2 FINITE ELEMENT FORMULATION

For a structure of arbitrary geometry, such as that shown in Figure 7.4, the equilibrium equations are as stated in Eqs 7.2a, 7.2b, and 7.2c. In general, there is no analytical solution for these coupled differential equations except for structural components of simple geometry, such as simple beams. Over the past 40 years, finite element methods have become indispensable for solving general mechanics problems. The methods and their tools provide numerical solutions that are sufficiently accurate for support of design decision making regarding the structural integrity of a design.

As stated earlier, the finite element method starts with the principle of virtual work. For a general structural problem (refer to Figure 7.4), the virtual work of the boundary value problem can be defined, according to the principle of virtual work, as

$$\delta W^{\text{int}} = \delta W^{\text{ext}} \tag{7.21}$$

where δW^{int} is the virtual work done by the internal forces, or

$$\delta W^{\text{int}} = \int_{\Omega} \delta U d\Omega = \int_{\Omega} \delta\left(\frac{1}{2}\boldsymbol{\sigma}\cdot\boldsymbol{\varepsilon}\right) d\Omega = \int_{\Omega} \boldsymbol{\sigma}\cdot\delta\boldsymbol{\varepsilon} d\Omega \tag{7.22}$$

where U is the strain energy, and $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ are stress and strain tensors, respectively. Note that for a 2D structure of biaxial stress state, $\boldsymbol{\sigma} = [\sigma_x, \sigma_y, \tau_{xy}]^{\text{T}}$ and $\boldsymbol{\varepsilon} = [\varepsilon_x, \varepsilon_y, \gamma_{xy}]^{\text{T}}$.

δW^{ext} is the virtual work done by the externally applied forces, or

$$\delta W^{\text{ext}} = \int_{\Omega} \boldsymbol{f}\cdot\delta\boldsymbol{u} d\Omega + \int_{\Gamma_1} \boldsymbol{T}\cdot\delta\boldsymbol{u} d\Gamma \tag{7.23}$$

where $\boldsymbol{f}$ and $\boldsymbol{T}$ are the body force and external traction vectors, respectively, and $\delta\boldsymbol{u}$ is the vector of virtual displacements. Therefore, the principle of virtual work states

$$\int_{\Omega} \boldsymbol{\sigma}\cdot\delta\boldsymbol{\varepsilon} d\Omega = \int_{\Omega} \boldsymbol{f}\cdot\delta\boldsymbol{u} d\Omega + \int_{\Gamma_1} \boldsymbol{T}\cdot\delta\boldsymbol{u} d\Gamma, \quad \text{for all } \delta\boldsymbol{u} \in Z \tag{7.24}$$

where Z is a kinematically admissible virtual displacement space.

Equation 7.24 must be solved through two important discretizations, *domain* and *function*. The finite element method discretizes the structural domain into small pieces, where each piece is called a *finite element* (i.e., an element with finite size or finite number of the elements). The finite element boundary edges (for surface elements) or faces (for 3D solid elements) are usually straight or flat for linear elements. For higher-order elements the element boundary can be a curve that matches better with the structural boundary. End points of the finite element edges (or corner points of the element faces) are called *nodes*. The structural domain discretized into finite elements is called a *mesh*. The finite element mesh approximates the structural domain, as illustrated in Figure 7.9. Note that forces,

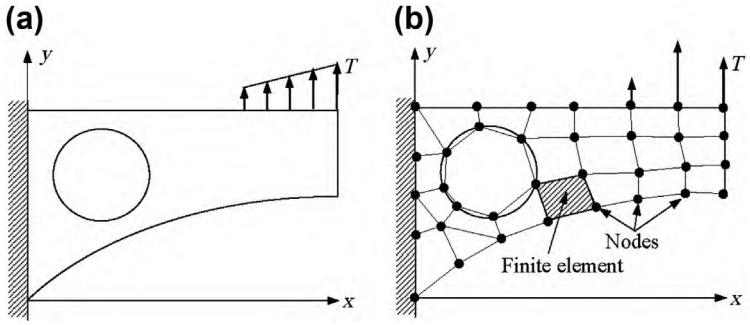


FIGURE 7.9

Domain discretization: (a) original structural domain and (b) finite element mesh.

such as the traction force T, are also discretized. The linearly distributed force T becomes equivalent to point forces applied at nodes.

Material properties, section properties, and distributed loads, in addition to the physical domain, are discretized in accordance with the finite element mesh. For example, a circular cross-sectional beam, as shown in Figure 7.10, is discretized into four finite elements. As a result, the cross-section and distributed load (and material property if applicable) are also discretized, most likely as average values within each respective element.

The second discretization is function discretization. Structural responses, such as displacements (or stresses), within the finite element are interpolated using the responses at the finite element nodes and element shape functions (or interpolation functions), which are usually polynomial functions. As illustrated in Figure 7.11, the interpolated solutions usually do not match with the exact solutions. The discrepancy is called *interpolation error*.

Displacements at the finite element nodes are called *degrees of freedom* (DOF). For a 2D planar problem, there are two DOF at each node, u_x and u_y. Consider a domain in a state of equilibrium discretized by a four-node quadrilateral finite element mesh, as depicted in Figure 7.12. According to the finite element method, the coordinates $\boldsymbol{x} = [x,y]^{\mathrm{T}}$ are interpolated from the nodal values $\boldsymbol{X}_j = [X_j,Y_j]^{\mathrm{T}}$:

$$\boldsymbol{x} = \sum_{j=1}^{4} \boldsymbol{N}_j \boldsymbol{X}_j = \sum_{j=1}^{4} \begin{bmatrix} N_j & 0 \\ 0 & N_j \end{bmatrix} \begin{bmatrix} X_j \\ Y_j \end{bmatrix} \tag{7.25}$$

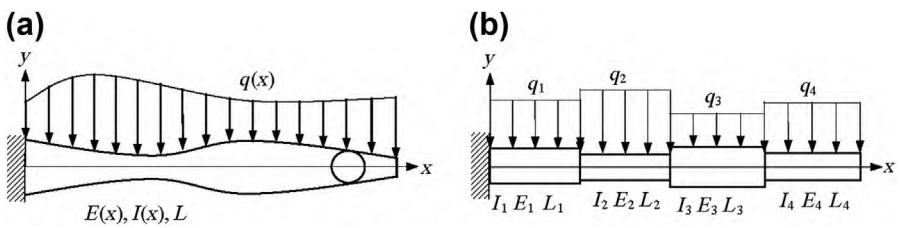


FIGURE 7.10

Domain discretization using a beam example: (a) physical problem and (b) finite element discretization.

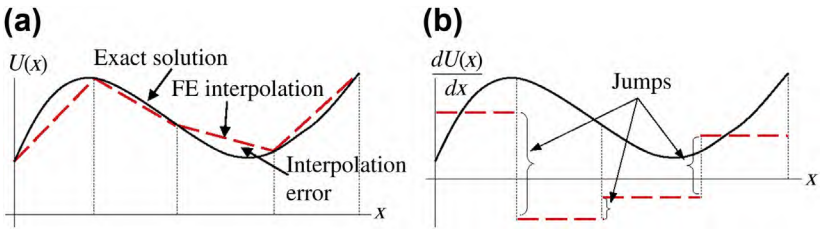


FIGURE 7.11

Function interpolation: (a) a piecewise linear interpolation for $U(x)$ and (b) derivative of $U(x)$ and jumps across the element boundary.

where $\boldsymbol{N}_j$ is the matrix of finite element shape functions for the jth node. For an isoparametric linear element, as shown in Figure 7.12, the shape functions are

$$N_1 = \frac{1}{4}(1-\xi)(1-\eta), \quad N_2 = \frac{1}{4}(1+\xi)(1-\eta)$$
$$N_3 = \frac{1}{4}(1+\xi)(1+\eta), \quad N_4 = \frac{1}{4}(1-\xi)(1+\eta) \tag{7.26}$$

in which ξ and η are the natural coordinates of the element. For an isoparametric finite element, displacement fields $\boldsymbol{u} = [u_x, u_y]^{\mathrm{T}}$ are similarly interpolated using the element shape functions and the nodal displacement values $\boldsymbol{U} = [U_x, U_y]^{\mathrm{T}}$:

$$\boldsymbol{u} = \sum_{j=1}^{4} \boldsymbol{N}_j \boldsymbol{U}_j = \sum_{j=1}^{4} \begin{bmatrix} N_j & 0 \\ 0 & N_j \end{bmatrix} \begin{bmatrix} U_{xj} \\ U_{yj} \end{bmatrix}$$

The strain field is computed directly from Eq. 7.3:

$$\boldsymbol{\varepsilon} = \sum_{j=1}^{4} \boldsymbol{B}_j \boldsymbol{U}_j \tag{7.28}$$

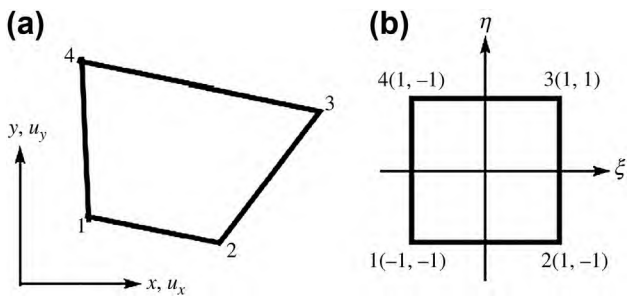


FIGURE 7.12

Two-dimensional isoparametric finite element of four nodes, (a) finite element, and (b) natural coordinates.

where the matrix $\boldsymbol{B}_j$ is defined in terms of derivatives of the shape functions $\boldsymbol{N}_j$:

$$\boldsymbol{B}_j = \begin{bmatrix} \dfrac{\partial N_j}{\partial x} & 0 \\ 0 & \dfrac{\partial N_j}{\partial y} \\ \dfrac{\partial N_j}{\partial y} & \dfrac{\partial N_j}{\partial x} \end{bmatrix} \tag{7.29}$$

and the chain rule is invoked to determine the coefficients of $\boldsymbol{B}_j$:

$$\begin{bmatrix} \dfrac{\partial \boldsymbol{N}}{\partial x} \\ \dfrac{\partial \boldsymbol{N}}{\partial y} \end{bmatrix} = \boldsymbol{J}^{-1} \begin{bmatrix} \dfrac{\partial \boldsymbol{N}}{\partial \xi} \\ \dfrac{\partial \boldsymbol{N}}{\partial \eta} \end{bmatrix} \tag{7.30}$$

where $\boldsymbol{J}$ is the Jacobian matrix,

$$\boldsymbol{J} = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} \tag{7.31}$$

Also, $\boldsymbol{\sigma} = \boldsymbol{D}\boldsymbol{\varepsilon}$, where $\boldsymbol{D}$ is the material stress-strain matrix or constitutive matrix. For a 2D plane stress problem, the matrix $\boldsymbol{D}$ is

$$\boldsymbol{D} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \dfrac{1-\nu}{2} \end{bmatrix} \tag{7.32}$$

If we insert the stress $\boldsymbol{\sigma}$ and strain $\boldsymbol{\varepsilon}$ into the left side of Eq. 7.24 and integrate over an element domain Ω_e, we have

$$\int_{\Omega_e} \boldsymbol{\sigma} \cdot \delta\boldsymbol{\varepsilon} d\Omega = \int_{\Omega_e} \delta\boldsymbol{\varepsilon}^{\mathrm{T}} \boldsymbol{D}\boldsymbol{\varepsilon} d\Omega = \delta \boldsymbol{U}^{\mathrm{T}} \int_{\Omega_e} \boldsymbol{B}^{\mathrm{T}} \boldsymbol{D}\boldsymbol{B} d\Omega \boldsymbol{U} = \delta \boldsymbol{U}^{\mathrm{T}} \boldsymbol{K}_e \boldsymbol{U} \tag{7.33}$$

where $\boldsymbol{K}_e$ is the stiffness matrix of an element Ω_e:

$$\boldsymbol{K}_e = \int_{\Omega_e} \boldsymbol{B}^{\mathrm{T}} \boldsymbol{D}\boldsymbol{B} d\Omega \tag{7.34}$$

Similarly, the right side of Eq. 7.24 can be discretized using element shape functions and expressed in an element as

$$\int_{\Omega_e} \boldsymbol{f} \cdot \delta \boldsymbol{u} d\Omega + \int_{\Gamma_{e1}} \boldsymbol{f}^{\mathbf{T}} \cdot \delta \boldsymbol{u} d\Gamma = \delta \boldsymbol{U}^{\mathrm{T}} \boldsymbol{F}_e \tag{7.35}$$

where $\boldsymbol{F}_e$ is the element force vector.

The element stiffness matrix $\boldsymbol{K}_e$ and the force vector $\boldsymbol{F}_e$ are assembled with those of all the other elements to form the global stiffness matrix $\boldsymbol{K}$ and the force vector $\boldsymbol{F}$. After imposing the displacement boundary conditions, the reduced matrix equations can be obtained as

$$\boldsymbol{K}\boldsymbol{U} = \boldsymbol{F} \tag{7.36}$$

This matrix equation is then solved numerically for nodal displacements and later for stresses.

As can be seen in the previous discussion, by going through the domain discretization and response interpolations, the partial differential equations that govern the continuum mechanics problem are converted into a system of linear (matrix) equations. The size of a matrix equation is determined by the degrees of freedom at the finite element node and the number of nodes created in the finite element mesh.

Interpolating displacement fields guarantees that the finite element solutions of the displacement responses are C^0-continuous, as depicted in Figure 7.11(a) using a 1D problem for simplicity, which is desirable. However, one major deficiency of this displacement-based formulation is that the stress (and strain) solutions, obtained by taking the derivatives of the displacement solutions, result in jumps at the element boundary. This jump can be seen clearly in Figure 7.11(b) if we go back to take the derivative of $U(x)$ in Figure 7.11(a). In the finite element method, stress jumps appear at element boundary points for 1D elements, element boundary edges for 2D elements, and element boundary faces for 3D elements.

Assuming the displacement solution shown in Figure 7.13(a), the difference between the true solution u^{ex} and the finite element solution is apparent. For a linear element (one with linear shape functions), the stress solution from FEA is $\widehat{\sigma} = E\frac{du}{dx}$, as depicted in Figure 7.13(b). To obtain a better stress field, a nodal averaging process is often employed to obtain the averaged nodal stresses $\overline{\boldsymbol{\sigma}}$; the displacement shape functions $\boldsymbol{N}$ are used again to interpolate a new stress field $\sigma* = \boldsymbol{N}\overline{\boldsymbol{\sigma}}$. It is apparent that $\sigma*$ provides a continuous stress solution across element boundaries, which is obviously a better approximation than $\widehat{\sigma}$. The difference between $\sigma*$ and $\widehat{\sigma}$ can be a good error estimate in stress.

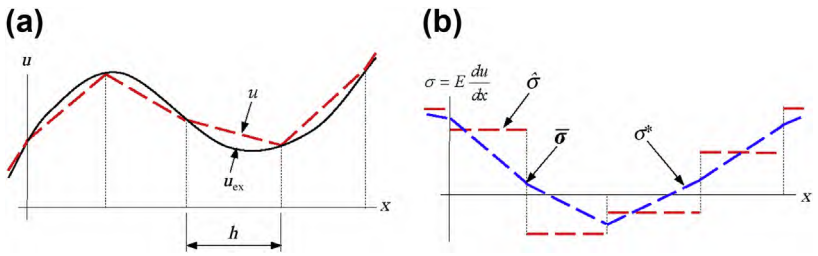


FIGURE 7.13

Stress jumps: (a) displacement fields and (b) stress functions.

A stress error e_σ can be estimated by

$$e_\sigma = \sigma^* - \widehat{\sigma} \tag{7.37}$$

Note that for 1D elements there is only one stress component: axial stress σ_x. Therefore, the stress error e_σ is a scalar function. For 2D or 3D elements, there are more stress components, where $\boldsymbol{e}_\sigma$ is a vector function. In general, the error estimate of the ith element can be evaluated in the energy norm as

$$\|\boldsymbol{e}_\sigma\|_i = \left(\int_{\Omega_{e_i}} \boldsymbol{e}_\sigma^T \boldsymbol{D}^{-1} \boldsymbol{e}_\sigma d\Omega \right)^{1/2} = \left(\int_{\Omega_{e_i}} \boldsymbol{e}_\varepsilon^T \boldsymbol{D} \boldsymbol{e}_\varepsilon d\Omega \right)^{1/2} \tag{7.38}$$

where

$\boldsymbol{D}$ is the constitutive matrix
$\boldsymbol{e}_\varepsilon$ is the vector of strain component error
$\boldsymbol{e}_\sigma$ is the vector of stress component error interpolated using element shape functions.

The global error estimate can be computed by summing $\|\boldsymbol{e}_\sigma\|_i$ over the entire finite element domain Ω, using

$$\|\boldsymbol{e}_\sigma\| = \left(\sum_{i=1}^{NE} \|\boldsymbol{e}_\sigma\|_i^2 \right)^{1/2} \tag{7.39}$$

where NE is the total number of elements. This estimate can be an effective indicator in determining if the current mesh is adequate. If $\|\boldsymbol{e}_\sigma\|$ is greater than a prescribed limit, the mesh needs to be refined. In that case, the element error estimate $\|\boldsymbol{e}_\sigma\|_i$ provides an excellent guide in terms of which area or elements need refinement.

It is obvious from Figure 7.13 that the error estimate of Eq. 7.39 is reduced if more elements are employed for analysis. In general, a finite element model with smaller element sizes generates more elements and therefore more nodes. As a result, the model has a larger set of equations to solve, which usually leads to more accurate solutions but longer computation time. Increasing the number of elements or refining the finite element mesh to achieve more accurate solutions is referred to as the *h-adaptation* ("*h*" denotes element physical size). In h-version FEA, a solution is converged by refining the finite element mesh and solving a larger set of equations, thus consuming more computation resources. Therefore, we have to invest wisely. If there is no error analysis capability such as Eq. 7.39 offered in the finite element code, as a rule of thumb we use smaller elements at high-stress areas, and larger elements at low-stress areas.

Several general-purpose h-version FEA codes are commercially available, such as ANSYS® (www.ansys.com), MSC/Nastran® (www.mscsoftware.com), and ABAQUS® (www.3ds.com). They all provide excellent finite element modeling and solution capabilities and are capable of solving general structural problems. For example, a 2D engine connecting rod, shown in Figure 7.14(a) (Edke and Chang, 2010), was modeled and solved using ANSYS, where about 500 linear isoparametric elements were employed to yield about 6,000 degrees of freedom. A set of firing loads were applied at the inner boundary of the connecting rod where the crankshaft and piston pin are

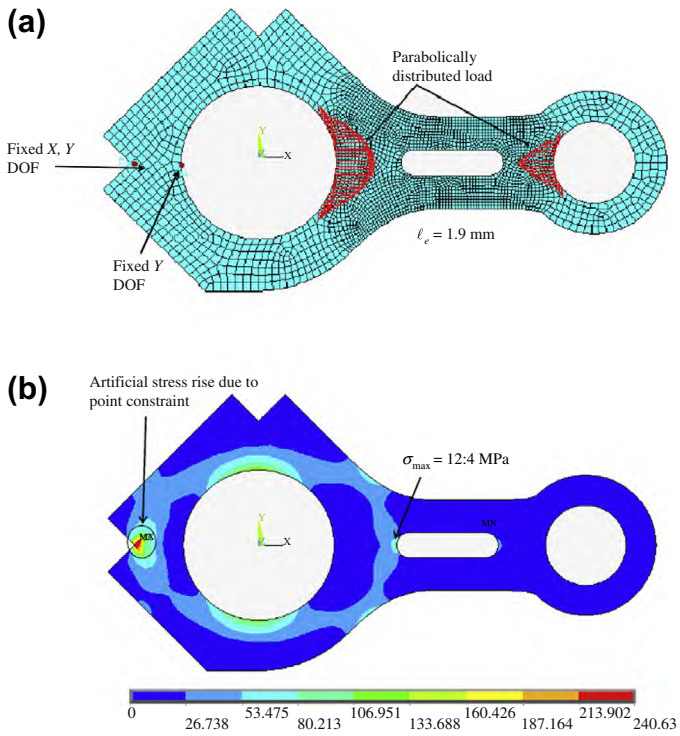


FIGURE 7.14

Two-dimensional engine connecting rod: (a) analysis problem with finite element mesh and (b) stress results obtained from ANSYS using linear isoparametric finite elements.

located. Boundary conditions were applied as suggested by Hwang et al. (1997). The maximum principal stress distribution in the connecting rod is shown in Figure 7.14(b). Although the maximum stress appears to be at the fixed node on the left side, it is merely an artificial stress concentration due to displacement constraints. The real maximum stress of $\sigma_{\max} = 124$ MPa occurs on the left semicircular edge of the slot.

In addition to the 2D isoparametric finite elements employed for the connecting rod example, there are many more element types available in commercial FEA codes. In general for h-version FEA, element shape functions are usually up to second order. A list of common finite element types supported in these codes is given in Figure 7.15.

7.3.3 *p*-VERSION FEA

The *h*-FEA achieves solution convergence by refining element size while retaining the polynomial order of the element shape functions (usually at lower order: $p = 1$ or $p = 2$); the *p*-FEA increases the polynomial order of the element shape functions to achieve solution convergence while maintaining the same finite element mesh. This concept is illustrated in Figure 7.16, where the exact solution u^{ex} is

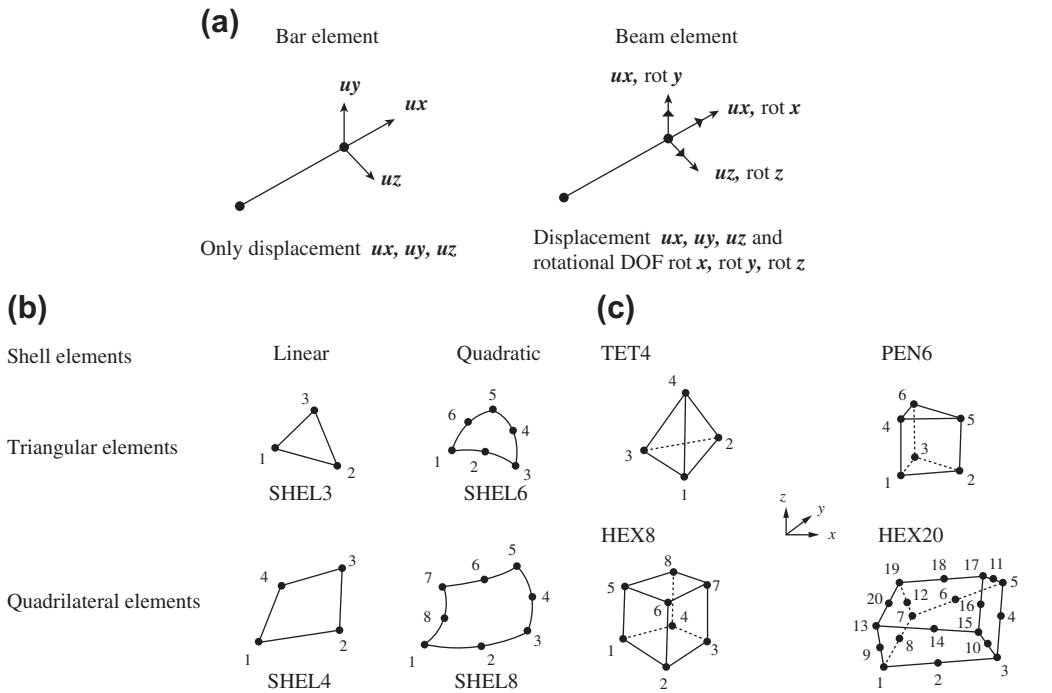


FIGURE 7.15

Typical finite element types: (a) bars and beams; (b) 2D shell elements; (c) 3D solid elements.

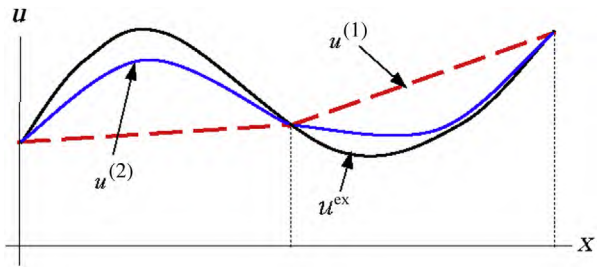


FIGURE 7.16

Displacement interpolations for $p = 1$ and $p = 2$.

approximated using a piecewise linear function $u^{(1)}$ ($p = 1$) and a piecewise quadratic function $u^{(2)}$ ($p = 2$). It is obvious that increasing the polynomial order of the interpolation function improves the accuracy of the approximation. Note that the polynomial order can reach up to $p = 9$ in some commercial p-FEA software, such as Pro/MECHANICA Structure (www.ptc.com).

One obvious advantage of the *p*-FEA is that the mesh does not have to be refined or adjusted for solution convergence. While increasing the polynomial order of element shape functions, a larger set of equations requires solution, which in general leads to more accurate solutions using more computational resources. Another advantage is that higher-order functions can be employed to represent element boundaries, yielding a much better approximation in geometry for general structures.

Research related to the *p*-version finite element method (*p*-FEM) has been ongoing since the early 1970s. *p*-FEM technology has been shown to be robust and superior to the conventional h-version FEM for important classes of problems, including nonlinear applications.

One of the important advantages of *p*-FEM over *h*-FEM is that it makes ensuring the quality of the computed information more efficient and more convenient. There are various measures of quality of approximate solutions. One is the energy norm. By definition, the energy norm is the square root of the strain energy, which is similar to the root-mean-square measure of error in stresses, as defined in Eq. 7.39. In engineering computations we are interested in structural responses, such as maximum normal stress, maximum von Mises stress, maximum displacement, and the first few natural frequencies. These are typical results offered by finite element solutions. It is important to know whether the finite element solutions are sufficiently close to the true (or exact) solution. Since generally we do not know the exact solution, this appears to be an unsolvable problem. However, we can always increase the polynomial order and rerun the analysis to achieve a better accuracy. While we are increasing the polynomial order for a series of analyses, if the difference in solutions obtained from the current polynomial order ($p = n$) and previous polynomial order ($p = n - 1$) is smaller than a prescribed tolerance, we consider the solution to converge to an acceptable value and so the solution process stops.

In *p*-version FEA, the stiffness matrix of the structure is formed by a set of hierarchical shape functions. "Hierarchical" means that when the polynomial degree increases from p to $p + 1$, the shape functions used in the polynomial degree p are not altered; in other words, the shape function set of polynomial degree p is a subset of that of polynomial degree $p + 1$. Methods for proper selection of the shape function set for *p*-version finite elements were addressed by Babuška et al. (1989). Since *p*-version FEA uses hierarchical shape functions, the data of the original computation in the lower polynomial element—portions of stiffness matrices, loading vectors, and so forth—can still be used for the higher-order element.

A human maxillary second molar is presented to demonstrate the advantages of using p-FEA to capture the geometry of the critical dentino–enamel junction (DEJ) and reach solution convergence. Creating an accurate geometry model for the tooth is challenging. The geometric modeling started with a histological section preparation of a human tooth. By tracing outlines of the tooth on the sections, discrete points were obtained and employed to construct B-spline curves that represent the exterior contours and DEJ of the tooth using least-square curve-fitting technique. The surface-skinning technique was then employed to quilt the B-spline curves to create a smooth boundary and DEJ of the tooth using B-spline surfaces. These surfaces were respectively imported into SolidWorks (www.solidworks.com) via its application protocol interface (API) to create solid models, as shown in Figures 2.37(c) and (d).

The solid models were then imported into Pro/MECHANICA Structure for finite element meshing and analysis. The finite element mesh shown in Figure 7.17(a) was created manually in Pro/MECHANICA Structure. The automatic mesh generation capability provided in

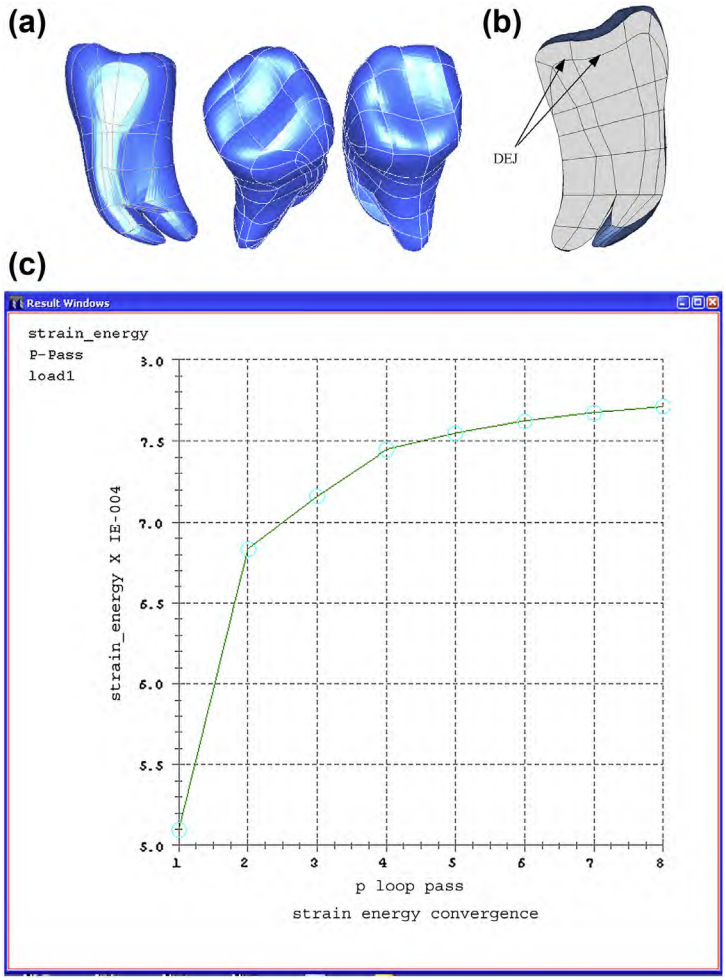


FIGURE 7.17

Tooth finite element model: (a) mesh, (b) section view with DEJ, and (c) convergence graph.

Pro/MECHANICA Structure was not employed because of its limitation on maintaining mesh continuity across the DEJ. The finite element mesh was created by splitting and merging surfaces and curves into desired segments. Then smaller solid volumes were created by choosing wrapping surfaces that form individual airtight cavities. Note that C^0-continuity had to be maintained to ensure validity of the finite element mesh since neither gap nor penetration between elements was allowed.

One finite element was assigned to each solid volume. All the elements in the model were 3D solid elements. This was the semiautomatic meshing capability offered by Pro/MECHANICA Structure.

Note that the DEJ was well preserved in the finite element model, as shown in Figure 7.17(b). There are 192 solid elements in this model. According to the literature (e.g., Middleton et al., 1990), a uniformly distributed vertical load of 170 N was applied on top of the tooth model. The entire exterior boundary 2 mm below the DEJ was fixed. The isotropic material properties, Young's modulus, and Poisson's ratio, assigned to the enamel and dentin were based on literature designating 8.5×10^4 MPa and 0.33 for enamel, and 1.98×10^4 MPa and 0.31 for dentin, respectively.

A finite element analysis of the solid tooth model was conducted using Pro/MECHANICA Structure. A linear static analysis was carried out for the loading, material, and constraint conditions described earlier. A 0.5% convergence in strain energy was defined as the criterion for ensuring solution convergence. The analysis took eight passes to reach convergent solutions, where the maximum polynomial order of element shape function was 8, and 40,059 linear equations were solved simultaneously in FEA. The convergence study ensured the accuracy of the FEA solutions. The convergence graph for strain energy is shown in Figure 7.17(c). The strain energy curve became almost flat at passes 7 and 8, indicating that no significant improvement would be achieved by further increasing polynomial order. From the mathematical theory of the finite element method, the strain energy of the finite element solution converged to the exact solution from below, as revealed in the convergence graph in Figure 7.17(c). The stress fringe plots shown in Figure 7.18(a) indicate that a stress concentration of 24 MPa occurs at the left exterior surface of the tooth. This is due to the asymmetry of the tooth geometry. The stress concentration derives mainly from the combination of axial and bending stresses in compression. In addition, the stress distribution along the DEJ, which largely determines the strength of the tooth, can be identified in Figure 7.18(b) (Chang et al., 2003).

In Pro/MECHANICA Structure and other FEA software, the "cut-plane" option is available for visualizing the FEA results interior to the object. Using this option, the interior stress distributions are revealed. Note that in Figure 7.19 the cut-planes were created for sections with 5% depth apart. A total of 18 sections were cut for visualizing the stress distribution. As shown in Figure 7.19, no significant stress jump appears in any of the sections, which demonstrates that the *p*-FEA is more accurate and suitable for tooth mechanics study.

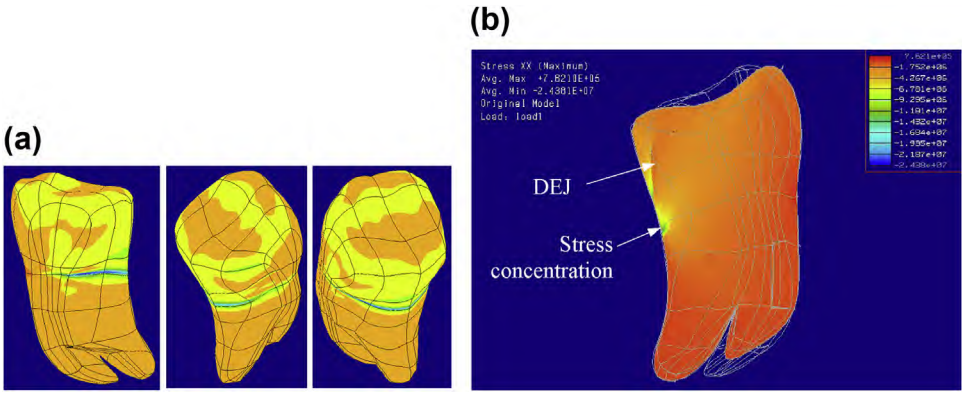


FIGURE 7.18

Tooth stress distribution in Pro/MECHANICA: (a) complete fringe plot and (b) sectional stress distribution.

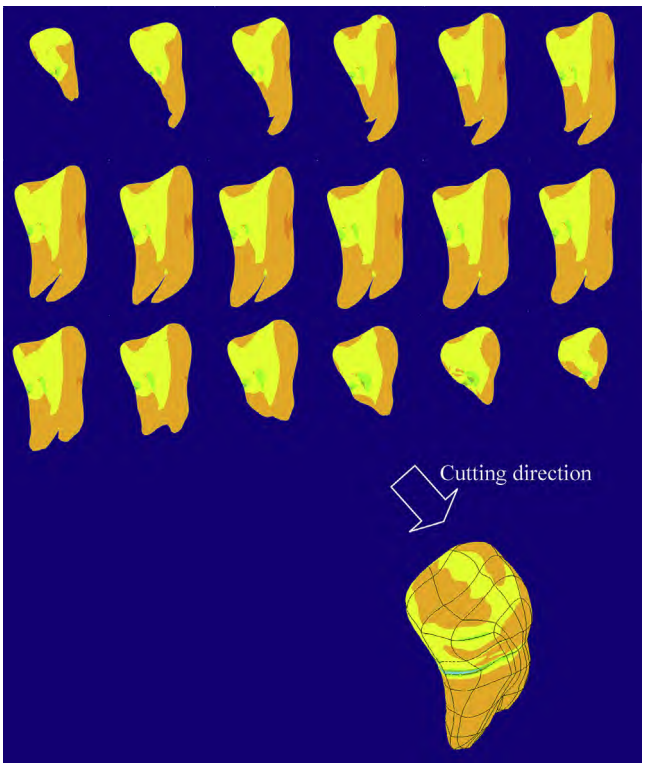


FIGURE 7.19

Stress fringe plots on cut-planes (in depth increments of 5%).

7.3.4 THE MESHLESS METHOD

A number of meshless (or mesh-free) methods that do not require an explicit mesh for finite element formulation have been developed recently. Smooth particle hydrodynamics (SPH) (Monaghan, 1988) was the first such method developed for handling infinite-domain astronomy problems. Others include the diffuse-element method (DEM) (e.g., Nayroles et al., 1992), the element-free Galerkin (EFG) method (e.g., Belytschko et al., 1994), and the reproducing kernel particle method (RKPM) (e.g., Chen et al., 1997).

The meshless method is appealing for solving problems in large-deformation analysis and structural shape optimization, where finite element mesh tends to become distorted (if regular FEA is employed), aborting analysis or optimization iterations. In this section, the RKPM will be briefly described, focusing only on the displacement interpolation using kernel functions.

In RKPM, the displacement field is interpolated globally as

$$u(\boldsymbol{x}) = \sum_{I=1}^{np} \Psi_I(\boldsymbol{x}) d_I \tag{7.40}$$

where

$\boldsymbol{x}$ represents a point in the structural domain
np is the total number of particles of the entire structure
$\Psi_I(\boldsymbol{x})$ is the RKPM interpolation function
d_I is the generalized displacement at the Ith particle.

Equation 7.40 reveals the very basic feature of RKPM (and the meshless method in general): The displacement field is interpolated in the entire structural domain (instead of a finite element) through the RKPM interpolation function $\Psi_I(\boldsymbol{x})$. In theory, then, no mesh is required.

Because of the global interpolation function $\Psi_I(\boldsymbol{x})$, more computational effort than that from FEM is expected. This function consists of a kernel function $\Phi_a(\boldsymbol{x}; \boldsymbol{x}_I - \boldsymbol{x})$ with a support measure of a (and a correction function for meeting the reproducing conditions). Note that $\boldsymbol{x}_I$ is the position vector of the Ith particle. A typical kernel function for planar problems, a bi-cubic B-spline surface, is shown in Figure 7.20(a).

The cubic B-spline kernel function shown in Figure 7.20(b) can be employed to interpolate a displacement function $u(x)$, shown in Figure 7.21. It is obvious that solution convergence can be achieved by refining the support size a or by increasing the number of particles; this is similar to h-p adaptation in mesh-based FEM.

The superiority of the RKPM in solving large-deformation problems has been demonstrated and widely recognized. Figure 7.22(a) demonstrates an application of the meshless method for calculating

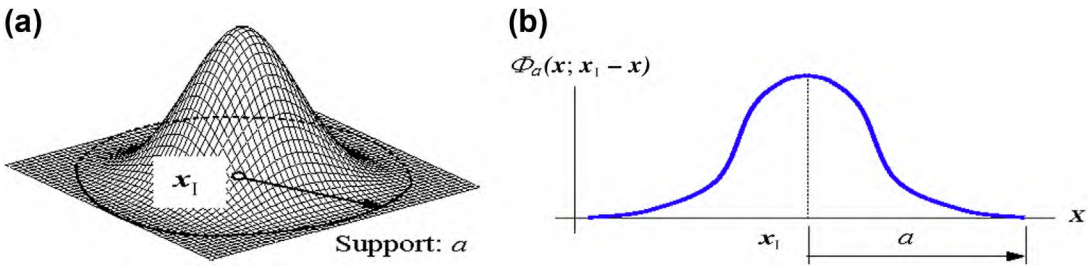


FIGURE 7.20

B-spline kernel function: (a) bi-cubic for planar problems and (b) cubic for 1D problems.

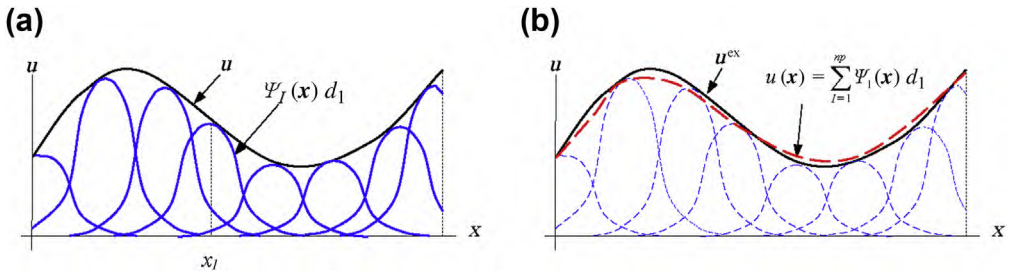


FIGURE 7.21

Displacement interpolations: (a) kernel functions and (b) summation of individual functions.

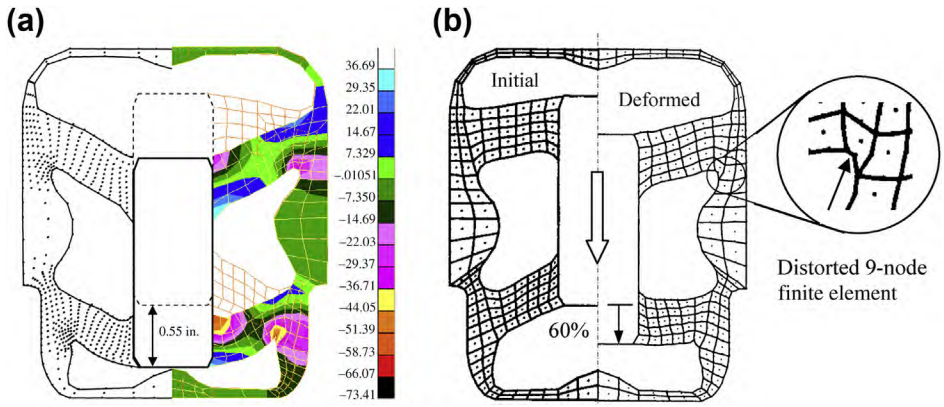


FIGURE 7.22

Engine mount: (a) meshless method and (b) regular finite element method.

the displacement and stress of an engine mount. The core metal bar can move down to the end (total travel is 0.55 in.) without any mesh-related problem. However, with the regular finite element method, the element mesh is distorted, aborting the analysis at 60% of the metal bar movement, as shown in Figure 7.22(b), where ABAQUS was employed for analysis (Grindeanu et al., 1998).

The RKPM method has been demonstrated to be feasible for solving large-deformation problems, for example, the engine mount problem shown in Figure 7.22. However, as illustrated in Figure 7.22(a) (see grids), a background mesh is still required for Gaussian integration in constructing a stiffness matrix. When the Jacobian of the background mesh becomes close to zero, Gaussian integration gives inaccurate results. Note that in general the geometry of the background mesh is much less restrictive than that of the shape of the true finite elements. The meshless method is an active research topic and is continuously being developed and enhanced. Another newly developed FEM, among others, is extended FEM (XFEM), which supports discontinuity problems, interface problems, and crack propagation problems, among others. Using XFEM for solving fracture mechanics problem will be briefly discussed in Section 9.6.

7.3.5 USING FINITE ELEMENT METHOD

In general, there are three major steps in using finite element method or FEA codes for solving structural problems. They are pre-processing, solutions, and post-processing, as depicted in Figure 7.23.

The pre-processing step starts with creation of a geometric model that represents the structural domain. The geometric model can be directly created using FEA codes with relatively limited modeling capability or, preferably, created in a CAD system and then exported to FEA codes. Once the geometric model is available, the finite element mesh, including nodes and elements, may be generated automatically, using the mesh generation capability in the FEA code, or manually, which is often less desirable, especially for structures of complex geometry. In addition, material properties and loading and displacement boundary conditions must be defined. Finally, a solution type (e.g., static, vibration, buckling, nonlinear) must be specified.

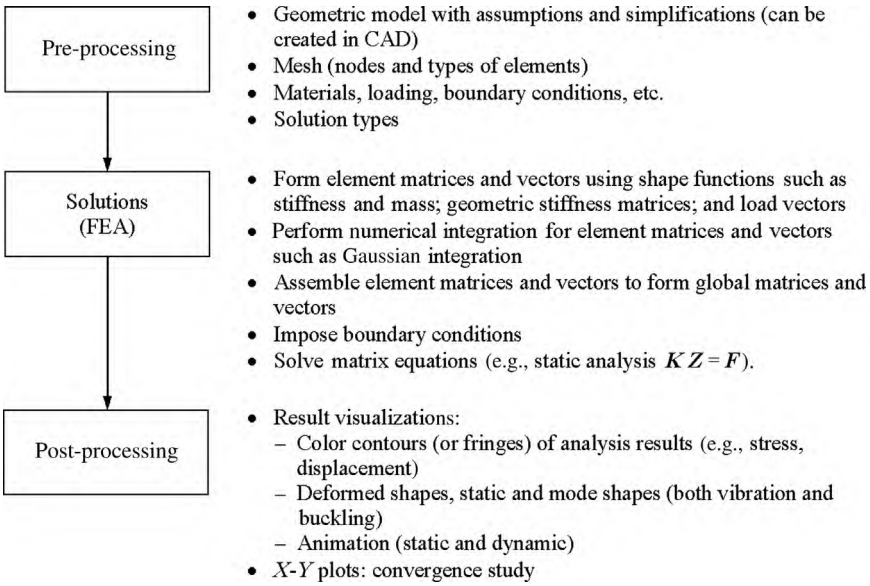


FIGURE 7.23

Overall process of structural analysis using finite element method.

The solutions step is dealt with by the FEA codes, which take the finite element model, formulate element matrices, assemble them for global matrices, impose boundary conditions, and solve the system of equations using numerical algorithms such as Gaussian elimination or LU decomposition (Atkinson, 1989), in which the stiffness matrix is decomposed and is the multiplication of two matrices, L and U (L is a lower triangular matrix and U is an upper triangular matrix).

The analysis results can be visualized in various forms in the post-processing step. Color fringe or contour plots of the solutions, such as displacement or stress, may be displayed; a deformed shape or animation may be requested to better visualize structural deformation. In addition, graphs for various results may be generated, such as the convergence study graph that was shown in Figure 7.17(c) if a convergence study was carried out using a p-version FEA.

7.4 FINITE ELEMENT MODELING

Since the first FEA computer code, Nastran, was developed in the late 1960s as a NASA initiative, tremendous advancement has been made for FEA, not only in theoretical development but, more importantly, in FEA software packages. Today commercial FEA codes are popular and widely accessible. Many mechanical engineers or engineering students can use FEA codes to solve structural problems with or without knowledge of fundamental FEA theory. FEA software is powerful and yet dangerous. This is because erroneous results due to mistakes made in creating FEA models can negatively impact design decision making in product development. It is extremely important for engineers to use FEA codes with competence, which is the main topic of this section.

We discuss the key elements of creating adequate finite element models, common pitfalls, commercial FEA packages, and strategies for solving complex problems. The elements to be discussed include model simplification and idealization, mesh generation, and boundary conditions. The goal is to provide readers with adequate information and knowledge to enable them to use FEA codes with sufficient confidence and competence for solving practical structural problems.

7.4.1 GENERAL PROCESS AND POTENTIAL PITFALLS

As shown in Figure 7.24, the first step in creating an FEA model is to fully understand the physical problem being considered. This includes the operating condition of the mechanical component or system, material properties, geometric shape and dimensions, boundary conditions, possible failure scenarios, and, most importantly, the questions FEA is to answer. Is it the maximum stress that could lead to material yield, the vibration frequency that could induce resonance, the load that poses a buckling hazard? We have to identify the structural performance or potential failure modes we want to monitor. This should determine the analysis type chosen in FEA.

The next step is about simplification and idealization of the structural problem. This is often necessary since the physical problems are usually too complex to solve as they are. A physical problem can be idealized by making adequate assumptions to reduce the complexity level to one that FEA is capable of solving, and yet with the numerical solutions closely resembling the behavior of the physical problem. These assumptions can be made by removing nonessential geometric features, reducing 3D structures to 2D or 1D components, converting impacts to equivalent static loads if feasible, fixing the rotating end of the structure for proper boundary conditions, and so forth. Moreover, the geometry of the structure may be simplified by taking advantage of the symmetric conditions, where only a portion of the structure is sufficient for solving the problem with the same accuracy. Idealization and simplification are two of the most challenging steps in conducting FEA. We will discuss this topic further in Section 7.4.2.

It is important to understand the physics of the structural problem to be solved and to identify the proper mathematical model that can be applied to solve the problem. There are at least three aspects to consider. The first is to understand in general how the deformation will mostly occur in the structure. Is it going to be a small deformation within the linear elastic range? Or, if the deformation is hyperelastic or plastic, is it beyond the linear elastic range? The second aspect is the load: Is it static or time-dependent? Third, is it a multiphysics problem, such as aero-structure, fluid structure, acoustics,

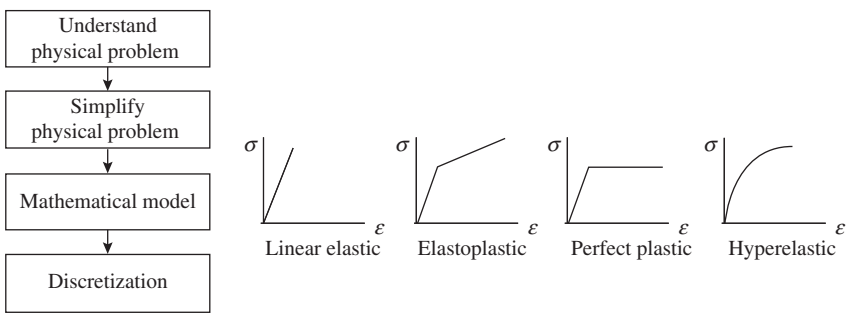


FIGURE 7.24

General process of creating FEA models.

or thermal structure? All of these must be well understood before creating finite element models and choosing an adequate FEA code for solution.

The final step involves finite element mesh generation. The questions to ask are: Does the problem require a very refined mesh, are the FEA software and computers able to handle a large finite element model? What type of elements will be used, linear or quadratic? Is p-version FEA a better choice for the problem at hand? If so, what will be the convergence criteria? Also, does the problem involve discontinuity in responses? These are essential questions to go over before creating the model. The more thought that is given to these questions, the better the preparation, which usually means a better FEA model and eventually time savings. Among the most important issues is what questions the FEA is to answer. These questions have to be very specific to avoid unnecessary or repeated effort in carrying out FEA.

7.4.2 IDEALIZATION AND SIMPLIFICATION

In engineering analysis, a physical problem often has to be idealized and simplified before it can be solved, especially with FEA. Idealization is about making proper assumptions so that the problem can fit into an existing mathematical model for solutions. One important aspect of idealization is reducing the complexity of the structural geometry in order to reduce the sizing of the FEA models yet maintaining solution accuracy. Simplification is about further reducing the size of the FEA models without compromising the accuracy in solutions. Idealization and simplification are probably the most important and yet challenging steps in creating FEA models. It takes time and experience to become competent in making these decisions. We will mention a few cases and draw some conclusions on this subject.

When working in CAD, especially in creating detailed design, the components being designed usually come with subtle geometry details. Very often it is difficult to mesh the entire model with the presence of small geometric features such as holes and fillets. Most of these small features can be suppressed or removed for finite element analysis since very often they do not affect the analysis results. Figure 7.25 illustrates a few such cases. However, features that affect the solutions cannot be ignored. For example, the small hole at the center of a plate under axial or bending load cannot be removed because the presence of the hole induces stress concentrations around it. In general, small features should be removed except for those that cause stress concentrations.

Another category in geometric idealization is reducing 3D solid components to surface or even line models to save computational effort. In general, a thin-shelled structure in 3D can be reduced to a shell surface model with its thickness entered as an element property in FEA. We may compress the outer and inner surfaces of the 3D thin-shelled solid model and use the compressed mid-surface as the shell surface for analysis, as shown in Figure 7.25(c). Such a capability is available in both Pro/MECHANICA and SolidWorks Simulation. A 3D solid beam like that shown in Figure 7.25(d) can also be reduced to a 1D line model with element section properties, such as cross-sectional area and moment of inertia, and beam orientation, such as the third node, entered for the beam elements in FEA.

The purpose of geometric idealization is mainly to cut down the size of the finite element model. Results obtained from FEA for the original and reduced models differ, but the difference is insignificant if the idealization is properly made. Geometric simplification takes advantage of structural symmetry to reduce model size. It is different from idealization since simplification does not affect the analysis results but cuts down computation time. To take advantage of symmetry in geometric

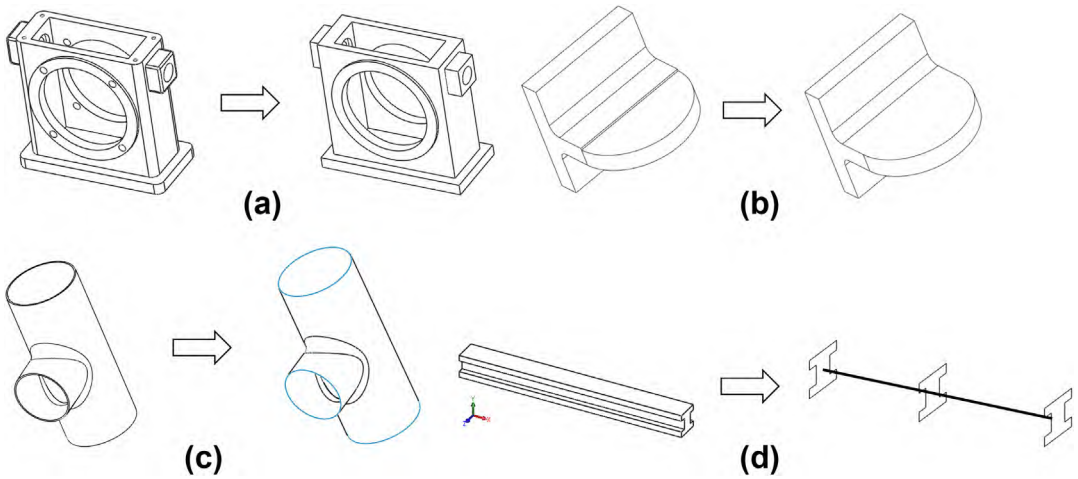


FIGURE 7.25

Geometric idealizations: (a) removing unnecessary features; (b) aligning surfaces; (c) reducing a 3D solid model to a surface or shell model (mid-surface); (d) reducing a 3D solid problem (beam) to a line model or frame structure (1D beam with cross-section properties, such as area, moment of inertia).

simplification, the structural geometry must be symmetric. The loading, boundary conditions, material properties, and geometric properties (thickness, beam section properties) must also be symmetric. Moreover, additional boundary conditions may have to be imposed on the simplified geometry to ensure symmetry of the structure in responses, such as the deformed shape.

For example, the rectangular plate shown in Figure 7.26(a) is loaded with two point loads horizontally, constrained at the left edge. This structure is symmetric with respect to the x-axis, assuming

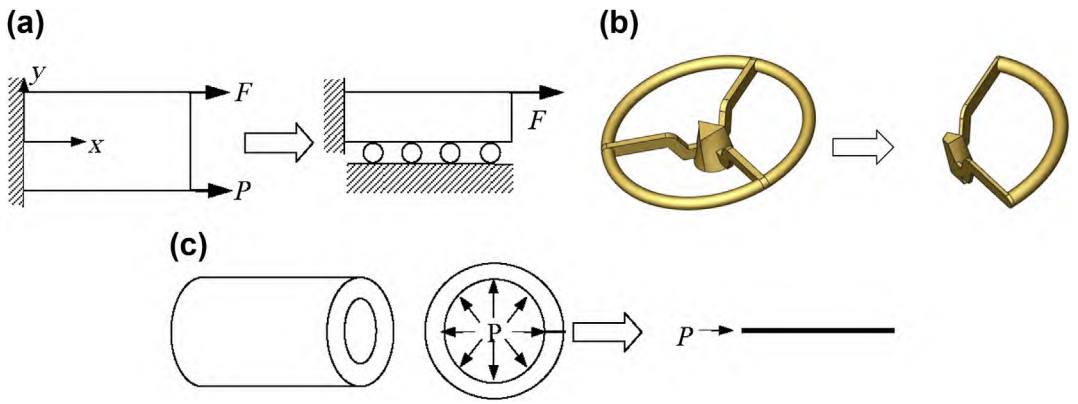


FIGURE 7.26

Geometry simplifications: (a) a rectangular plate, (b) an axisymmetric disk reduced to symmetrical portion, and (c) a long tube.

that the material is isotropic and homogeneous, that the two point forces F and P are identical, and that the thickness of the plate is uniform. Therefore, the plate can be split into two halves and only the upper (or lower) half analyzed, as shown. To impose the symmetric conditions, roller boundary conditions are added to the split boundary. The roller conditions ensure that the material on the middle axis only moves horizontally and that the material will not deform to cross the middle axis. The condition imposed is simply $u_y = 0$ along the x-axis. Solving half of the structure is much more efficient than solving the whole model. More important, the solutions for the half-structure are in theory identical to those of the full structure. The numerical solutions from FEA show very slight differences due to truncations in numerical computation.

Similarly, an axisymmetric problem, shown in Figure 7.26(b), can be reduced to a smaller domain (in this case 1/3 of the whole structure) if load, boundary conditions, material, and section properties are also axisymmetric. In this case, it is much easier to impose the symmetric boundary conditions at the split faces using polar coordinate systems. Moreover, sufficiently long tubing with internal pressure P may be reduced to a 1D problem, as shown in Figure 7.26(c).

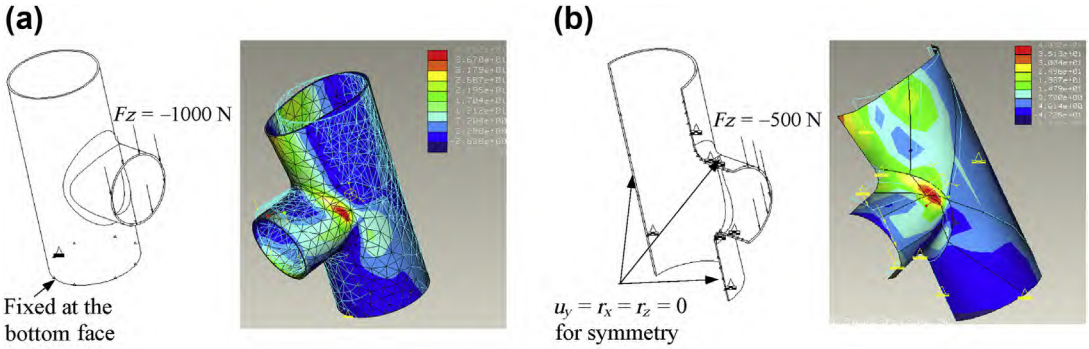
The thin-walled tank example in Figure 7.27 is employed to further illustrate the idea of idealization and simplification. A full 3D solid model was first created and meshed for FEA, where 3,585 tetrahedron elements were generated using AutoGen (automatic mesh generator) in Pro/MECHANICA. A 1% strain energy convergence criterion was employed for solution convergence. As shown in Figure 7.27(c), the number of equations solved in the final pass of the p-solution was 143,343. The maximum principal stress and displacement magnitude were 41.6 MPa and 0.427 mm, respectively.

The tank was cut in half due to symmetry. With half of the load and symmetric boundary conditions imposed, the half-tank was further reduced to the surface model, shown in Figure 7.27(b). This surface model was meshed into 12 shell elements. The same convergence criterion was defined. The solution in strain energy converged in nine passes. There were 2,733 equations solved in the final pass (versus 143,343 equations in the full 3D solid model). The maximum principal stress and displacement magnitude were 40.2 MPa and 0.428 mm, respectively, which were very close to the principal stress and displacement of the full 3D solid model. However, as shown in Figure 7.27(c), the CPU time required for the simplified model was only about 3% that of the full 3D model (3.25 out of 106 seconds). The savings in computation time due to model simplification and idealization were significant, as this example demonstrates.

Some FEA codes, including SolidWorks Simulation, offer geometric idealization to treat structural joints, such as pins, bolts, and rigid joints. Special mathematical models serve to replace the detailed geometry and finite element mesh of such connectors, greatly simplifying the analysis of structural assemblies. In Simulation, a 3D beam assembly can be easily reduced to frame structures with truss or beam elements, in which joints are automatically created to replace the connection in solid form between components. In addition to treating joints, Simulation provides contact capabilities, where components in an assembly are allowed to come into contact without penetrating each other.

7.4.3 MESH GENERATION AND REFINEMENT

Mesh generation generates a polygonal or polyhedral mesh that approximates the geometric domain of the structure. This is a key step in finite element analysis. A given structural domain must be partitioned into simple elements meeting in well-defined ways. First, elements must be continuous or



(c)

Model	*Model A: Full solid model*	*Model B: Half-surface model*
Geometry	Solid	Surface
Number of finite elements	3,585 solid elements	12 shell elements
Number of equations in last pass	143,343	2,733
Number of passes	4	9
Maximum edge order	5	9
Strain energy convergence	<1%	<1%
Maximum displacement magnitude	0.4274 mm	0.4277 mm
Maximum principal stress	41.62 MPa	40.22 MPa
Total elapsed time*	826.02 seconds	11.89 seconds
Total CPU time*	106.02 seconds	3.25 seconds
Maximum memory used	346,574 KB	166,614 KB
Working directory disk usage	649,216 KB	4,096 KB

*Dell Precision PWS670, Intel Xeon 2.80 GHz 2.00 GB RAM

FIGURE 7.27

Thin-walled tank problem: (a) full 3D solid model, (b) half-surface model, and (c) performance comparison between these two models.

conformal, which means that they all have to share common element edges and corner nodes, as shown in Figure 7.28(a). No elements should penetrate or overlap with any other. Second, all elements should be well shaped, which generally involves restrictions on the edge angles or aspect ratio of the elements (Figure 7.28(b)). For example, the best-shaped quadrilateral (or quad) element is square, and the best-shaped triangular element is an equilateral triangle. However, in practice, square and equilateral elements are difficult to retain because of irregular structural boundaries. The mesh quality directly impacts the accuracy of the FEA as well as the computational resources involved. Finally, there should be as few elements as possible, but some portions of the domain may need smaller elements so that the solution is more accurate there.

Most FEA codes provide automatic mesh generation capabilities. For surface structures, triangular mesh is most common. Some codes also provide mesh with a mix of triangular and quadrilateral elements or even completely quadrilateral elements. For solid structures, only tetrahedral elements are currently supported by almost all commercial FEA software. Most provide some mesh control and refinement options. Certain ranges for angle and aspect ratio may be specified to ensure mesh quality.

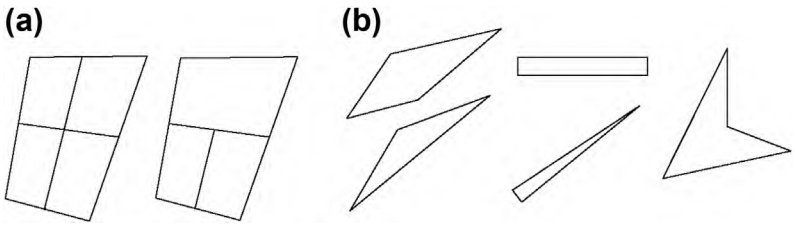


FIGURE 7.28

Finite element mesh: (a) conformal (*left*) and nonconformal (*right*) meshes; (b) poorly shaped elements: excessive angles (*left*), excessive aspect ratio (*center*), and negative Jacobian (*right*).

In some cases, global and local element sizes can be specified separately. In addition, a face, an edge, or points for local mesh refinement can be chosen.

Some FEA pre-processors, such as HyperMesh® (www.altairhyperworks.com) and MSC/Patran® (www.mscsoftware.com), offer semiautomatic mesh generation capabilities. These cases call for manually splitting the structural domain into continuously decomposed domains (the splits are called patches or volumes for 2D and 3D domains, respectively), and then creating mesh in these decomposed domains according to a mapping method to be discussed later. This approach provides flexibility in decomposing the structural domain and allows the user to create all quad or all hexahedral elements. The semiautomatic approach offers more flexibility in choosing types of finite elements, but requires much more manual effort.

7.4.3.1 Automatic Mesh Generation

In general, mesh generation technology deals with structured and unstructured meshing. Structured meshing is commonly referred to as "grid generation." Strictly speaking, a structured mesh can be recognized by all interior nodes of the mesh having an equal number of adjacent elements. The mesh generated by a structured grid generator is typically all quad or all hexahedral.

Unstructured mesh generation, on the other hand, relaxes the node valence requirement, allowing any number of elements to meet at a single node. Triangular and tetrahedral meshes are most commonly thought of when referring to unstructured meshing, although the quadrilateral and hexahedral meshes can be unstructured as well.

Most FEA codes employ methods based on unstructured mesh technology. Triangular and tetrahedral are by far the most common forms of unstructured mesh generation. Most techniques currently in use for support of triangular or tetrahedral meshing can fit into one of three main approaches: octree, Delaunay, and advancing front.

7.4.3.1.1 Triangular and Tetrahedral Meshes

The octree technique was primarily developed by Mark Shephard's group at Rensselaer Polytechnic Institute in the 1980s. In this method, cubes containing the geometric model are recursively subdivided until the desired resolution is reached. Figure 7.29(a) shows the equivalent two-dimensional octree model decomposition. Irregular cells are then created where cubes (squares for 2D model shown in Figure 7.29(a)) intersect the boundary, often requiring a significant number of geometric intersection calculations. Tetrahedra (or triangular elements for 2D models) are generated from the irregular cells on the boundary and the internal regular cells.

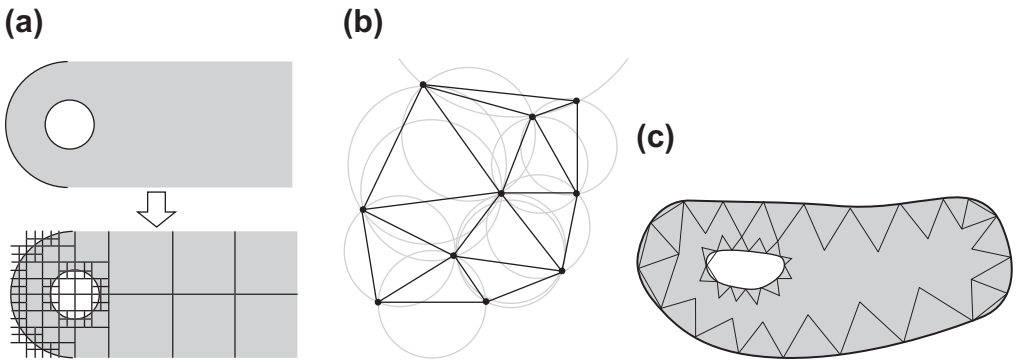


FIGURE 7.29

Meshing technology: (a) octree decomposition of a simple 2D object; (b) Delaunay triangulation in the plane with circumcircles shown; (c) advancing front, where one layer of triangles has been placed.

By far the most popular mesh generation techniques for triangular and tetrahedral meshing are those utilizing the Delaunay criterion, as illustrated in Figure 7.29(b). The Delaunay criterion, sometimes called the "empty sphere" property, states that a triangle net is a *Delaunay triangulation* if all the circumcircles of all the triangles in the net are *empty*. This is the original definition for a two-dimensional space. It is possible to use the criterion in a three-dimensional space if a circumscribed sphere replaces the circumcircle. A circumsphere can be defined as the sphere passing through all four vertices of a tetrahedron.

Another very popular family of triangular and tetrahedral mesh generation algorithms is represented by the advancing, or moving, front method. An active front is maintained where new tetrahedra are formed. Figure 7.29(c) is a simple two-dimensional example of the advancing front, where triangles have been formed at the boundary. As the algorithm progresses, the front advances to fill the remainder of the area with triangles. In three dimensions, for each triangular facet on the front, an ideal location for a new fourth node is computed. Also determined are any existing nodes on the front that may form a well-shaped tetrahedron with the facet. The algorithm selects either the new fourth node or an existing node to form the new tetrahedron based on which forms the best one. Also required are intersection checks to ensure that tetrahedron does not overlap as opposing fronts advance toward each other.

7.4.3.1.2 Quadrilateral Meshes

Unstructured quadrilateral (or quad) meshing algorithms can, in general, be grouped into two main approaches: direct and indirect. With an indirect approach, the domain is first meshed with triangles. Various algorithms are then employed to convert the triangles into quadrilaterals. One of the simplest methods for indirect quadrilateral mesh generation includes dividing all triangles into three quadrilaterals, as shown in Figure 7.30(a). A 3D tetrahedron element can also be split into four hexahedra, as shown in Figure 7.30(b). This method guarantees an all-quadrilateral mesh, but a high number of irregular nodes are introduced into the mesh, resulting in poor element quality. An alternate algorithm is to combine adjacent pairs of triangles to form a single quadrilateral, as shown in Figure 7.30(c). However, while the element quality is improved, a large number of triangles may be left.

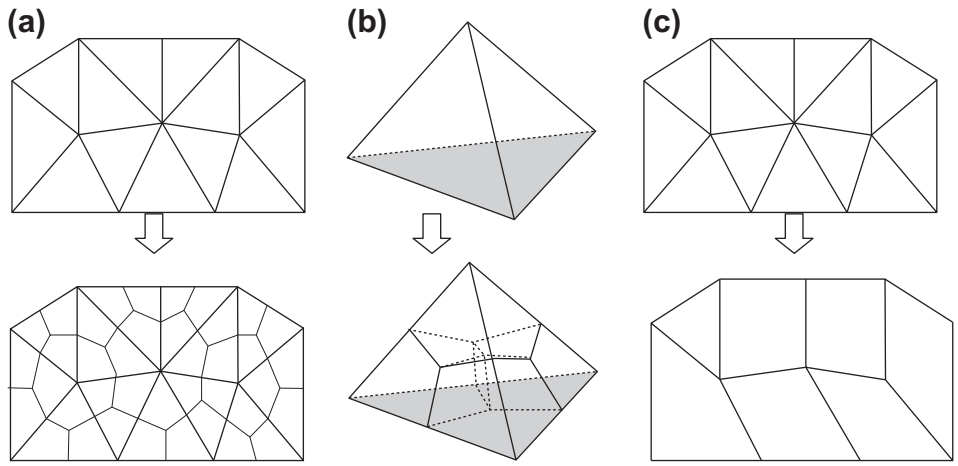


FIGURE 7.30

Indirect quad mesh: (a) a quad mesh generated by splitting each triangle into three quads through mid-point subdivision; (b) decomposition of a tetrahedron into four hexahedra; (c) a quad-dominant mesh generated by combining triangles.

Many methods for direct generation of quad meshes have been proposed. With a direct approach, quadrilaterals are placed on the surface directly, without first going through the process of trianglar meshing. Among these methods, there appear to be two types. The first comprises methods that rely on some form of decomposition of the domain into simpler regions that can then be meshed following standard approaches, such as the mapping method. The second comprises those that utilize a moving front for direct placement of nodes and elements.

One of the most advanced decomposition methods is *medial axis transformation* (MAT). The medial axis (MA) is defined as the locus of centers of locally maximal circles inside an object. The locus with the radii of the associated locally maximal circles is defined as the object's medial axis transform (MAT). The boundary and the corresponding MA of a 2D object are shown in Figure 7.31(a). The points where three or more line entities of the MA meet are called branch points. Associated with the MA is a radius function R, which defines for each point on the axis its shortest distance to the object's boundary. At each branch point, line segments are created to connect the branch point to the tangent points of the circle with the boundary. These line segments, which decompose the physical domain of the structure, are called corridors, as illustrated in Figure 7.31(b). Having decomposed the area into simpler regions, sets of templates are then employed to insert quadrilaterals into the domain, as shown in Figure 7.31(c). A more recent method, called LayTrack, creates uniform spacing inside the corridors to generate uniform, isotropic quad-elements of good aspect ratio (for example, Quadros et al., 2004). A similar concept has been applied to 3D objects, where a medial surface is created by rolling maximal spheres inside an object, and then hexagon mesh is created, as depicted in Figure 7.31(d).

A representative method in the second kind of direct quad mesh generation is plastering, which is an extension of the advancing front algorithm to 3D objects. In this method, elements are first placed

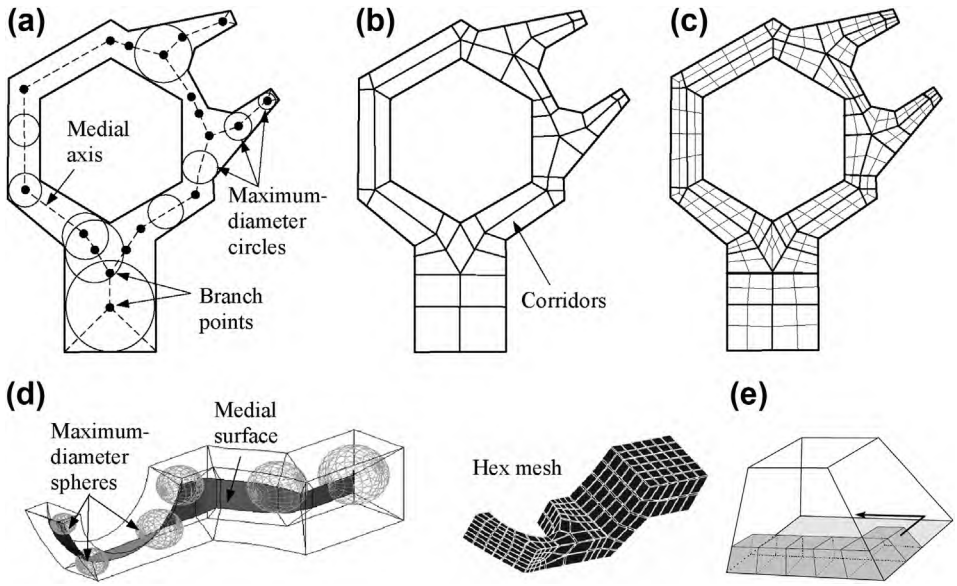


FIGURE 7.31

Direct methods for quad and hex meshes: (a) the medial axis of the domain; (b) decomposing domain into small corridors; (c) a quad mesh created in each corridors; (d) the medial surface and hex mesh for a 3D object; (e) the plastering process forming elements at the boundary.

starting with the boundaries and advancing toward the center of the volume, as shown in Figure 7.31(e). A heuristic set of procedures for determining the order of element formation is defined. Similar to other advancing front algorithms, a current front is defined consisting of all quadrilaterals. Individual quads are projected toward the interior of the volume to form hexahedra. In addition, plastering must detect intersecting faces and determine when and how to connect to preexisting nodes. As the algorithm advances, complex interior voids may result, which in some cases are impossible to fill with all-hex elements. Existing elements, already placed by the plastering algorithm, must sometimes be modified to facilitate placement of hexes toward the interior. The plastering algorithm has yet to be proven reliable on a large class of problems.

7.4.3.1.3 Mesh Improvement

It is uncommon for a mesh generation algorithm to be able to define a mesh that is optimal without some form of post-processing to improve the overall quality of the elements. The two main categories of mesh improvement include smoothing and clean-up. Most smoothing procedures involve an iterative process that repositions individual nodes to improve the local quality of the elements, such as edge angles and aspect ratios. Clean-up generally refers to any process that changes element connectivity.

From a practical perspective, almost all popular FEA codes provide automatic mesh generation capabilities that support tri- and quad-elements for shell and tetra-elements for solid structures. Ranges

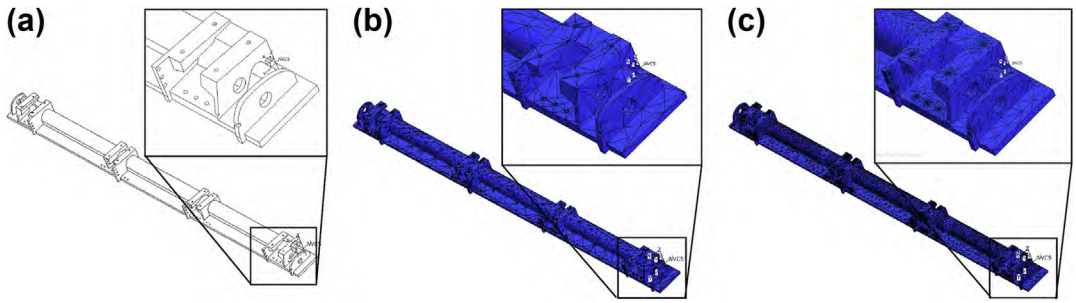


FIGURE 7.32

Three-dimensional torque tube: (a) CAD model, (b) first mesh of 8246 tetrahedral elements, and (c) second mesh of 63,816 tetrahedral elements.

of critical mesh parameters, such as aspect ratio and edge angles, can be defined up front. Also, local refinements are supported in most codes.

Here we present a 3D torque tube example to provide a better picture of the mesh generation capabilities offered by FEA codes. The torque tube shown in Figure 7.32 is created in Pro/ENGINEER. The length of the tube is about 70 in. Small features in the tube such as thin fins and small holes are shown in Figure 7.32(a).

The tube geometry is fairly complex. The AutoGen capability offered by Pro/MECHANICA Structure, a p-FEA program, was employed to create mesh for the tube. Default parameters for mesh generation, including allowed edge and face angles between 5 and 175 deg. and an allowed aspect ratio of 30, were employed; 8,246 tetra-elements were successfully created in about 90 seconds CPU time on a Pentium IV PC. As shown in Figure 7.32(b), the curve boundary was largely matched with the mesh; however, the mesh quality was less desirable because of the large aspect ratio (the resulting maximum aspect ratio was 11.3).

The default parameters were adjusted by reducing the aspect ratio to 10 and the angle ranges to 25 and 155 deg.; AutoGen was again employed. This time, 63,816 tetra-elements were successfully created in 201 minutes CPU time on a Pentium IV PC, 130 times more than the first case. As shown in Figure 7.32(c), the mesh quality was much improved (the resulting maximum aspect ratio was 4.40). The second mesh also captured the geometry around the holes much more accurately. Note that AutoGen does not always generate mesh successfully. With a smaller aspect ratio, it often fails. The best approach is to use default parameters to create a baseline mesh and then adjust mesh parameters for an improved mesh whenever possible.

7.4.3.2 Semiautomatic Mesh Generation

Although automated mesh generation methods for two- and three-dimensional structures have been studied intensively, many engineers still craft meshes manually for a certain class of analysis problems. One of the missing capabilities of current commercial mesh generators is the versatile control of mesh anisotropy and directionality.

In general, quad- and hex-elements are more desirable than tri- and tetra-elements. First, quad- and hex-elements use bilinear interpolation functions for the displacement field, providing more accurate

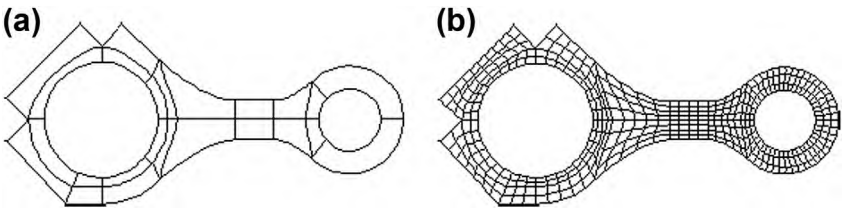


FIGURE 7.33

Two-dimensional connecting rod: (a) a Patran patch model and (b) a finite element mesh.

solutions than those of tri- and tetra-elements in general. As a result, it takes fewer quad- or hex-elements to achieve accurate solutions. For example, if the triangular element is used in shell analysis, its membrane behavior is very poor and inaccurate results will be obtained for many problems (Brauer, 1993; Wilson, 2000; Zienkiewicz, 1989). Also, it is reported in the literature that certain problems, such as plasticity and rubber analysis (Finney, 2001) and eigenvalues problems (Benzley et al., 1995) converge better with quad- or hex-elements.

When the automatic mesh generator at hand is not able to create the desired mesh or fails to generate one, the mesh has to be created manually. The key idea for manual mesh generation is to decompose the structural domain into smaller continuous regions and then use a mapping method to create the mesh. This is the semiautomatic approach. CAD and CAE packages, such as MSC/Patran (www.mscsoftware.com), HyperMesh, and ANSYS provide a mapping capability for mesh generation.

Two examples, a 2D connecting rod, shown in Figure 7.33, and a 3D turbine blade (see Figure 7.35), are employed to illustrate the semiautomatic mesh generation method. The domain of the connecting rod was first decomposed into 25 geometric patches in Patran, as shown in Figure 7.33(a). In Patran, a geometric patch is represented mathematically as a bi-cubic parametric surface (or the Coons patch discussed in Chapter 2), as depicted in Figure 7.34. Obviously, one geometric patch is not able to represent a complicated geometric boundary. Usually, a structural domain is decomposed into several smaller patches in the modeling process. The decomposed patches must be at least C^0-continuous to generate a conformal mesh. Also, to avoid sharp edge angles in finite element

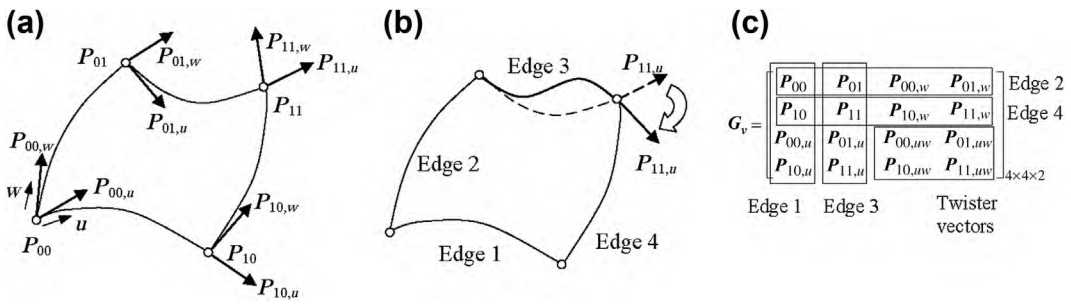


FIGURE 7.34

Bi-cubic parametric patch: (a) corner points and tangent vectors; (b) adjusting the tangent vector $\boldsymbol{P}_{11,u}$; (c) data that define the patch formulated in a 4×4×2 matrix.

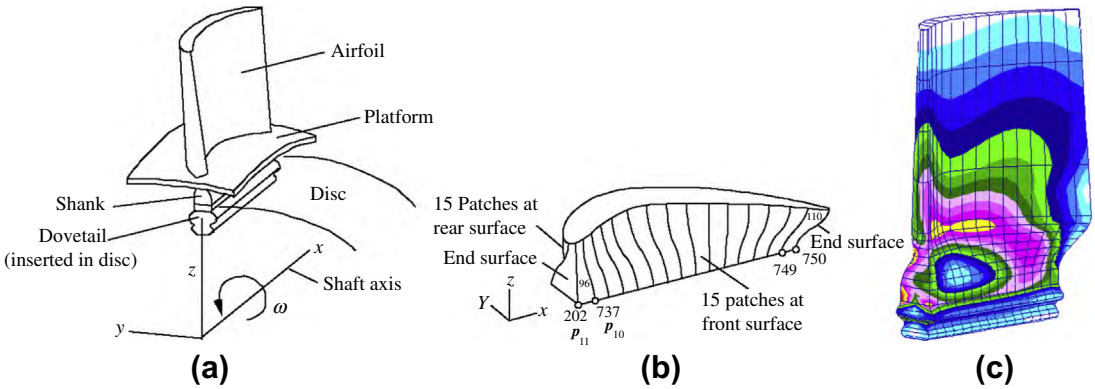


FIGURE 7.35

Three-dimensional turbine blade: (a) the physical model, (b) boundary patches created for the shank, and (c) the finite element model with a stress fringe plot.

mesh, the patch boundary edges must be properly created by adjusting the tangent vectors at the corner points.

A typical patch in Patran, shown in Figure 7.34, is a bi-cubic parametric surface in terms of parameters u and w, where u and $w \in [0,1]$. The patch edges are cubic curves in u or w. The shape of the cubic curve can be controlled by adjusting tangent vectors in both direction and magnitude. For example, changing the direction of the tangent vector $\boldsymbol{P}_{11,u}$ alters the geometry of Edge 3 and therefore the geometry of the patch, as illustrated in Figure 7.34(b). Aligning tangent vectors at the junction between patches ensures geometric smoothness across the patch boundary. A mesh of quad-elements can then be generated by specifying the number of elements along the u and w parametric directions, respectively, as shown in Figure 7.33(b). This is essentially the mapping method.

Physically, a turbine blade is inserted into a slot of a disc mounted on a rotating shaft, as shown in Figure 7.35(a). The blade can be divided into four parts: airfoil, platform, shank, and dovetail. When the turbine is operating, fluid pressure is applied to the surface of the airfoil and an inertia body force is applied to the whole blade structure because of rotation. To generate the mesh, the shank was first decomposed into 15 smaller pieces, represented by C^1-continuous patches at the front and rear surfaces as shown in Figure 7.35(b). Furthermore, the shank was decomposed in to 15 hyperpatches (volumes) using the respective front and rear patches. Hex-elements were then created in individual hyperpatches using the mapping method. The same procedure was employed to create mesh for the airfoil and dovetail, as shown in Figure 7.35(c).

7.4.4 CAD MODEL TRANSLATIONS

Creating a geometric model using finite element codes—or even an FEA pre-processor such as MSC/Patran—by dealing with points, curves, and patches is often tedious and time-consuming. It is much more convenient and productive to create a geometric model in CAD using feature-based solid modeling capabilities. Creating mesh using an automatic mesh generator in CAD is straightforward.

However, from time to time it may be necessary to bring the CAD model into finite element code for a more desired mesh, such as quad- or hex-elements.

Some FEA provides a direct connection to popular CAD software. For example, in ANSYS, you may directly import the Pro/ENGINEER solid model for mesh. Very often such direct connection is not available. In that case, two options are commonly employed: IGES (initial graphics exchange system) (see Stokes, 1995) and STEP (standard for the exchange of product model data), as discussed in Chapter 6.

7.4.5 LOADS AND BOUNDARY CONDITIONS

One of the most challenging tasks in creating finite element models is dealing with boundary and loading conditions. The finite element model must be properly restrained by displacement constraints to eliminate rigid-body motion and, more important, to accurately capture physical conditions. In many cases, the components being modeled are assembled to other parts through bolts, pins, and so on. Very often these connectors link the structure to other components that are constrained physically. These constrained components are seldom included in FEA. It may not always be a good idea to constrain the nodes on the inner surface of the holes where pins or bolts are located, since they are not constrained physically. Unexpected stress concentration often appears around the hole where displacement constraints are placed.

There are several ways to work around this problem. First, semi-cylinders may be added to fill half of the holes where pins or bolts reside, and the displacement constraints placed at the center axis of the semi-cylinders, as shown in the clevis example in Figure 7.36(a). Similarly, forces are applied at the center axis of the semi-cylinders at the other end. As depicted in Figure 7.36(a), there are stress concentrations close to the center axes where displacement constraints and forces are placed. However, these high stresses can be ignored since they are artificial. The real stress concentrations are properly captured at the top concave surfaces, as shown in Figure 7.36(a).

Another approach is to create beams to connect the nodes at the inner surface of the holes and constrain the other end of the beams instead of constraining the nodes on the inner hole surface, as shown in Figure 7.36(b). In the roadarm example, two elastic beams are added to the left hole that represents the torsion bar to which the roadarm is mounted. The torsion bar is constrained to better resemble physical conditions. At the lower end of the torsion bar, numerous beams of greater stiffness (or even rigid beams) are added to connect to the nodes at the inner surface of the hole. As a result, there is no stress concentration around the hole, which again better resembles physical situations.

Similar to displacement constraints, application of loads to finite element models is critical. In many cases, loads are not stationary. Either a structural dynamic analysis must be conducted or the load must be converted to stationary for static analysis. Also, an isolated point load is fine for a beam structure but not for 2D or 3D solids since an isolated point load tends to create stress concentrations. Physically, applying a point load to a structure is like pushing a needle into an elastic object, which yields high stress concentration and is not physically meaningful. In general, a distributed pressure load is more realistic.

If an isolated point load is unavoidable, a better approach is to add beams to diffuse the force flow, similar to that employed for displacement constraints, as shown in the roadarm example. As shown in Figure 7.36(b), an elastic beam is added to the hole on the right to simulate the axis of the roadwheel. At the lower end of the beam, a point force is applied that simulates the force acting on the roadarm from the roadwheel. At the upper end of the beam, numerous rigid beams are added to connect the end

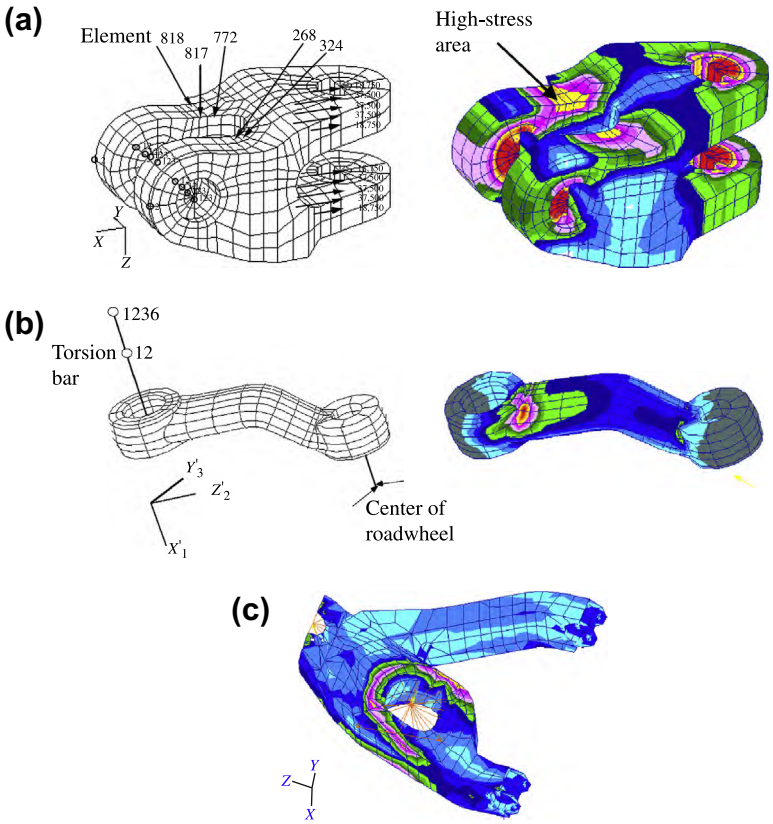


FIGURE 7.36

Creating proper boundary and loading conditions: (a) tracked vehicle clevis, (b) tracked vehicle roadarm, and (c) lower control arm of an HMMWV.

to the nodes at the inner surface of the hole. As shown in Figure 7.36(b), high stress concentration is properly captured at the top surface closer to the left hole.

A similar example, shown in Figure 7.36(c), involves a lower control arm of a high-mobility multipurpose wheeled vehicle (HMMWV). Again, numerous rigid beams are added to connect the nodes at the inner surface of the large hole. At the other end of the beams, a point load is applied to simulate the force delivered by the shock absorber to the control arm. Similar treatment can be seen at the left end in order to add a point load that simulates the force acting from the ball joint. As shown in Figure 7.36(c), stress concentration is properly captured. No artificial stress concentrations reveal.

7.4.6 RESULTS CHECKING

Finite element codes are powerful and capable of solving general structural problems. However, these codes are dangerous. Mistakes made in the finite element model yield erroneous results, which may put

the product design in jeopardy if they are not carefully checked and corrected. It is important to note that no software tools are error-proof. Only limited warnings and error checking are provided, so it is absolutely critical for engineers to find every possible way to verify the FEA results, either by hand calculations (which are not always possible) or by reviewing the FEA results and exercising educated engineering judgment. Good engineers make sound judgments; incompetent engineers live with fuzzy guesses.

It is good practice to predict, or at least have some rough idea about the FEA results beforehand. After running FEA, if the FEA results deviate significantly from prediction, it is either that the prediction is wrong or that the FEA results are incorrect (due to mostly modeling errors). In either case, this presents a golden opportunity to learn and improve our design engineering skill.

The first thing to check is the unit system. CAD software usually offers choices in unit systems. Once chosen, the system must be followed consistently in creating the FEA model. For those who are familiar with Pro/ENGINEER, the default unit system is in.-lb$_m$-sec—that is, inch in length, pound mass in mass, and second in time. The question is, what is the difference between pound mass and pound force (the latter being more familiar)? This can be a pitfall if care is not taken.[1]

A very simple cantilever beam is discussed to further illustrate the danger of ignorance, especially regarding the unit system. In this example the default unit system in.-lb$_m$-sec in Pro/ENGINEER is chosen. The beam has a 1 in. by 1 in. square cross-section and a 10 in. length. The left end of the beam is fixed and a distributed load of 1,000 is added to the top edge of the front end face, as shown in Figure 7.37(a). The material is aluminum 2014 with properties 4.09E+09 and 0.33 for Young's modulus and Poisson's ratio, respectively, given by Pro/MECHANICA. The beam is meshed into 12 tetra-elements (Figure 7.37(a)).

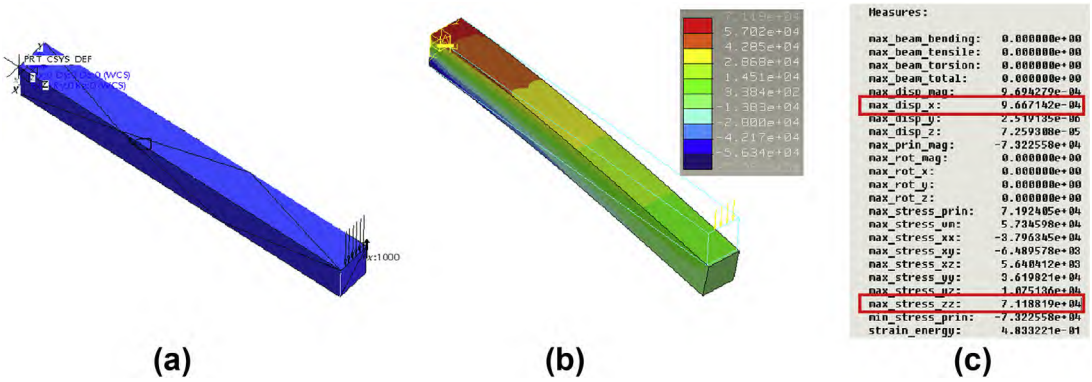


FIGURE 7.37

Cantilever beam: (a) boundary and load conditions with mesh, (b) deformed shape with bending stress fringe plot, and (c) results of default measures.

[1]CNN reported on September 30, 1999, that NASA lost a $125 million Mars orbiter because a Lockheed Martin engineering team used English imperial units of measurement while the agency's team used the more conventional metric system for a key spacecraft operation.

The FEA results obtained by a single-pass run in Pro/MECHANICA show that the maximum bending stresses of 71,188 occur at the top and bottom faces (tension and compression, respectively, close to the left end), as shown in Figure 7.37(c). The deformed shape in Figure 7.37(b) looks fine. The issue is whether the FEA results are correct and what these numbers mean. The best way to resolve the issue is to calculate the bending stress using classical beam theory from strength of materials. According to the theory, the maximum bending stress is

$$\sigma_b = \frac{Mc}{I} = \frac{(F \times L)(h/2)}{bh^3/12} = \frac{(1000 \times 10)(1/2)}{1 \times 1^3/12} = 60{,}000 \tag{7.41}$$

where F, L, b, and h are force, length, width, and height of the beam cross-section, respectively. The theory says that the bending stress is 60,000, but the FEA results indicate a value of 71,188—a 30% difference, which is significant for this simple example. Which one is correct? Which one should be trusted?

In fact, both are correct in their respective context. The difference is largely due to Poisson's ratio, which classical beam theory does not take into consideration. In FEA, we entered 0.33 for the ratio and expected to see lateral shrinkage when the transverse load was applied, which contributed partially to the stress along the longitudinal direction. That is why the bending stress was higher than calculated. FEA certainly provides more detailed and realistic results. If Poisson's ratio is set to zero and the analysis is rerun, the bending stress becomes very close to 60,000. Note that in using other software (for example, SolidWorks Simulation), there may be slightly higher stresses at the corners of the beam cross-section at the root that are due to stress concentration. The question now is, what is the stress unit? Pro/MECHANICA indicates that the stress unit is $\text{lb}_m/(\text{in. sec}^2)$ instead of psi (lb per in.^{2}) that we are more familiar with. What is $\text{lb}_m/(\text{in. sec}^2)$? Will the beam yield or fracture at 60,000 $\text{lb}_m/(\text{in. sec}^2)$?

According to Newton's second law, the force unit in the default in.-lb_m-sec system is $\text{lb}_m\text{in./sec}^2$. As discussed in Appendix 7A, 1 lb_f = 386 $\text{lb}_m\text{in./sec}^2$, and the force in $\text{lb}_m\text{in./sec}^2$ units is 386 times smaller than that in lb_f. When you apply a 1 lb_f force to a 1 lb_m mass block, it accelerates 386 in/sec^2, whereas a 1 lb_m in./sec^2 force only accelerates the same block 1 in./sec^2.

In what units is the 1,000 force applied? In fact, the force applied is 1,000 lb_m in./sec^2, which is only about 2.5 lb_f. Such a small force causes a very small stress in the beam—that is, 60,000 $\text{lb}_m/(\text{in. sec}^2)$, which is only about 155 psi. This also explains why the vertical displacement is so small. From FEA, the vertical displacement is 9.667×10^{-4} in. This result can be verified using classical beam theory:

$$\delta = \frac{PL^3}{3EI} = \frac{1000 \times 10^3}{3(1.06 \times 10^9)(1 \times 1^3/12)} = 9.780 \times 10^{-4}\text{in.} \tag{7.42}$$

which is very close to that of FEA. Certainly this small displacement is due to a small 2.5 lb_f force.

Classical beam theory is in fact very useful in verifying FEA results. Many practical problems, even complex ones such as the roadarm in Figure 7.37(b), can be verified using it. Stress and displacement result verification for the roadarm is left as an exercise. However, there are still large portions of the physical problems that are beyond beam theory. When analytical solutions are out of the question, what can be done? First, the FEA model must be checked to make sure that there is no modeling error, by reviewing the deformed shape of the structure to make sure all displacement boundary conditions are satisfied, including symmetry conditions if applicable. Then how the loads are applied must be reviewed.

Do they resemble the physical conditions? Are there any potential artificial stress concentrations? Also, it must be ascertained that the material properties are correctly defined.

After the model is verified, the stress pattern should be reviewed to see if it is consistent with the load and boundary conditions defined. A convergence study may be necessary. Finally, educated engineering judgment must be exercised to check the numerical results. For example, in the cantilever beam case, if the force unit was mistakenly understood as 1,000 lb_f, the question arises as to why such a large force produced only 9.667×10^{-4} in. of displacement on a 1 in. $\times$1 in. $\times$10 in. aluminum bar. Engineers must not be afraid to ask questions. When all are answered, the FEA results are most likely to be accurate, which creates greater confidence.

7.4.7 STRATEGY FOR COMPLEX PROBLEMS

Structural analysis problems are usually very complex. This complexity can be attributed to structural geometry, loading conditions, material behavior, and problem type. The most challenging part of dealing with a complex problem is not creating an FEA model for analysis but creating a model that correctly answers the questions asked at the outset.

The best strategy for tackling a complex problem is to start with a simple one. The problem must be simplified by making proper assumptions, hopefully to a point that the solutions of the simplified FEA model can be verified analytically and yet reasonably resemble the physical problem. Solutions of the simplified FEA model serve as a baseline for any improved results that follow.

The fidelity or efficacy of the simplified FEA model must be improved by removing or relaxing the assumptions. Any improvement increases the complexity of the problem and increases FEA computation time. One general strategy is to relax the assumptions that cause the least computation increment first. For example, we want to stay in a linear elastic range initially and gradually extend to nonlinear or plasticity, if applicable, which takes more analysis time. The FEA results obtained from a linear elastic model are most likely not what can be used. Very often this model serves well for model verification. Solutions to time-dependent problems, such as dynamics, impact load, or viscoelasticity, are probably the most time–consuming. Also, it is more efficient to start with a simplified geometry by suppressing small and insignificant features and with a relatively coarse mesh. The mesh may be gradually refined and the suppressed features brought back that can impact the results.

In any improvement, it is critical to check the FEA results and compare them with the baseline result before moving to the next improvement. A baseline can be stress or displacement, calculated using analytical equations or FEA models that are verified beforehand. Any significant deviations should raise a warning flag and merit careful rechecking. Mesh refinement and convergence studies are usually the final activities carried out to improve the accuracy of results.

Basically, we want to start with a simple problem, relax assumptions one or two at a time, check results using baseline or engineering judgment, and then iteratively increase the fidelity level of the FEA models until they closely reassemble the physical problem at hand and produce FEA results of acceptable accuracy and confidence.

7.5 COMMERCIAL FEA SOFTWARE

A large number of commercial FEA software tools are currently available. General-purpose codes support general applications, such as static, buckling, dynamic, and multiphysics. Some have strong

ties with CAD through designated interfaces; some are even embedded in CAD. There are also special-purpose codes, such as LS-DYNA®, for nonlinear structural dynamic simulation for problems such as crashworthiness. In this section, a brief overview of commercially available FEA codes is presented, including their strengths and weaknesses.

7.5.1 GENERAL-PURPOSE CODES

There are several general-purpose FEA codes embedded in their respective CAD systems. These are SolidWorks Simulation (called COSMOSWorks® before 2009) in SolidWorks; Pro/MECHANICA Structure in Pro/ENGINEER; CATIA® FEA in CATIA V5 (www.3ds.com); and FEA in NX. All provide a seamless connection between CAD and FEA without the need for a geometric translator. All user interactions are performed within the CAD environment, including pre-process, analysis, and post-process. The learning curve for these software tools is usually less steep if the user has prior experience with the respective CAD systems.

All of the codes support basic FEA problems, including static, vibration, and buckling. All are primarily h-version, except for Pro/MECHANICA, which is a p-version FEA. SolidWorks Simulation supports nonlinear and fatigue analysis. Pro/MESH, an h-FEA mesh generator in Pro/ENGINEER, supports both pre- and post-processing of commercial FEA codes. Pro/MESH automatically meshes Pro/ENGINEER parts, supports defining FEA models in Pro/ENGINEER, and supports a loose connection to major FEA codes (through ASCII input data files). Pro/ENGINEER also reads and displays FEA results. Meshing capability in Pro/MESH is somewhat limited, supporting only tetrahedral elements for 3D solid structures, and it sometimes produces undesirable meshes.

In addition to Pro/MECHANICA, another popular p-FEA is StressCheck™ (www.esrd.com), probably the most theoretically sound p-FEA code, which was developed by Barna Szabó at Washington University in St. Louis; it was later commercialized by Engineering Software Research and Development, Inc. StressCheck supports 2D and 3D models, linear and nonlinear problems, and fracture mechanics problems.

Three well-known and popular FEA codes—ANSYS, MSC/Nastran, and ABAQUS—offer more engineering capabilities and more options in finite element types. ANSYS was one of the earliest FEA codes and is probably the most popular code in academia. It is h-FEA and recently began offering p-elements. ANSYS supports fairly complete analysis capabilities for solving general engineering problems. Nastran is one of the best general-purpose FEA codes, providing probably the most complete set of capabilities for general engineering problems, including linear, nonlinear, elastic, plastic, aero-structure, acoustic, transient dynamics, and others. Nastran's quadrilateral plate element is considered to be the most robust in the field. MSC/Nastran supports mainly h-elements. It supports p-elements as well, but they are not its main capability. ABAQUS is one of the best commercial FEA codes for nonlinear analysis; its nonlinear hyperelastic and plastic capabilities are among the best. Recently ABAQUS incorporated XFEM for crack propagation simulation.

In terms of CAD connections, ANSYS offers direct connections to most major CAD packages, including Pro/ENGINEER, SolidWorks, CATIA, UG, SolidEdge® (www.plm.automation.siemens.com), and Autodesk® Inventor®, in addition to neutral formats such as IGES. On the other hand, MSC/Nastran ABAQUS interface with MSC/Patran for pre- and post-processing, especially in importing CAD geometry. In addition to MSC/Patran, HyperMesh from Altair® Engineering provides

excellent CAD interfaces as well as semiautomated mesh capabilities. It also offers excellent post-processing capabilities similar to those of Patran.

7.5.2 SPECIALIZED CODES

Specialized codes usually offer analysis capabilities geared to much focused engineering fields in addition to standard capabilities. One typical software considered by industry and academia to be the tool for crashworthiness analysis is LS-DYNA, which also offers excellent capabilities for solving nonlinear problems, including nonlinear dynamics, contact, crack propagation, and coupling problems, such as structural, fluid, thermal, and acoustics.

Another specialized code for nonlinear FEA is MSC.Marc®. Marc and Mentat combined deliver a complete solution (pre-processing, solution, and post-processing) for implicit nonlinear FEA. Marc is capable of reliably solving problems that involve changing contact conditions between components and/or large strain (plasticity or elastomeric behavior, for example), especially for rubber-related applications such as tires and seals. Mentat is the premier environment for understanding, exploring, and interacting with nonlinear analysis. It is tightly integrated with the MSC.Marc FEA program, allowing data to be defined interactively through a graphic user interface.

Another code worth mentioning is COMSOL Multiphysics® (www.comsol.com), a finite element analysis software package for various physics and engineering applications, especially coupled phenomena and multiphysics. COMSOL Multiphysics also offers an extensive interface to MATLAB® (www.mathworks.com) and its toolboxes for a large variety of programming, pre-processing, and post-processing possibilities.

7.6 CASE STUDY AND TUTORIAL EXAMPLES

Before wrapping up this chapter, we will go through a case study in which scanned anatomy images are used to create finite element models. In addition, tutorial examples, including a simple cantilever beam and a thin-shelled tube, are provided. Step-by-step instructions in creating these tutorial examples are given in the tutorial projects P3 or S3 of this book. Model files can be found on the companion website (http://booksite.elsevier.com/9780123820389).

7.6.1 CASE STUDY

In this case study, we present a practical and systematic method for constructing an accurate finite element model for a human middle ear (Sun et al., 2002). The middle ear is the portion of the ear internal to the eardrum and external to the oval window of the cochlea, as shown in Figure 7.38. It contains three ossicles (malleus, incus, and stapes), which transform vibrations of the eardrum into waves in the fluid and membranes of the inner ear. The hollow space of the middle ear is the tympanic cavity, or *cavum tympani*. The eustachian tube joins the tympanic cavity with the nasal cavity (nasopharynx), allowing pressure to equalize between the middle ear and the throat.

Middle ear components have a tiny and complex geometry, and they play an essential role in sound transmission. Sounds collected and conveyed in the external ear canal are first transformed into mechanical vibrations of the eardrum and ossicular chain, and then transformed into traveling waves in the fluid-filled cochlea (inner ear). Devices for hearing restoration in cases of middle ear conductive

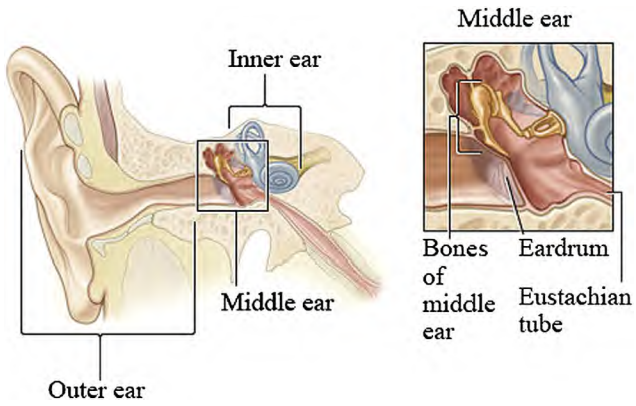


FIGURE 7.38

Human middle ear anatomy. (Source: (left) http://upload.wikimedia.org/wikipedia/commons/d/d2/Anatomy_of_the_Human_Ear.svg & (right) http://en.wikipedia.org/wiki/Ear#Middle_ear)

impairments (sensorineural loss) are often mechanical in nature and thus can be studied using mechanical computational models.

From a mechanics perspective, the middle ear is essentially an impedance match between the external ear and the inner ear. Its transfer function, related to the dynamic behavior of the ossicular chain, is complex because of its anatomy. The objective of this case study was to create an accurate three-dimensional (3D) geometric model of the human middle ear that allows observation of its 3D structure for morphological studies and for developing finite element models for mechanics analysis.

The modeling process started with extracting a normal fresh human temporal bone that was fixed, decalcified, and embedded in celloidin. The resulting block was removed from the chloroform, trimmed, mounted on a microtome plate, and stored in 70% alcohol. Before the celloidin block was sectioned, four parallel fiducial holes (perpendicular to the cutting plane) were made in it with a drill press. Permanent ink was injected into the fiducial holes to stain the marks. Sections were then cut on a sliding microtome at a thickness of 20 μm, as shown in Chapter 2, Figure 2.36(a).

Next, in sequence, every tenth section was stained and mounted on glass slides. Finally, each histology slide was scanned into a computer using a flatbed scanner and saved as an image file (Chapter 2, Figure 2.36(b)). The images were aligned with a template constructed by a typical section image using the fiducial marks. Aligned images were then trimmed as standard-sized images, brought onto a sketch plane in CAD software (in this case SolidWorks), and fitted in a reference rectangle of the same size as the image so that the actual size could be represented accurately. The images were then digitized by marking points along the outlines of the middle ear structures as identified by an otologic surgeon. These structures included eardrum, ossicles, attached ligaments, and muscle tendons (Chapter 2, Figure 2.36(c)). The B-spline curves were constructed to best fit these points using the curve-fitting technique (Chapter 2, Figure 2.36(d)) and were quilted to form the closed boundary surfaces for individual middle ear components using the surface-skinning technique (Tang and Chang, 2001). The solid models of the middle ear components were constructed using these surfaces and then assembled to create the entire middle ear model, as shown in Chapter 2, Figure 2.36(e).

All surfaces of the geometric model were translated into HyperMesh for finite element modeling. Based on these surfaces, the FE mesh of the middle ear components and attached ligaments/tendons was created with the HyperMesh meshing capabilities. Chapter 2, Figure 2.36(f) shows the anterior-medial view of the middle ear FE model with attached ligaments/tendons and cochlear fluid constraints. A total of 1,746 three-node triangular and four-node quadrilateral shell elements were created to mesh the eardrum. Surrounding the eardrum periphery, the tympanic annulus was modeled using 113 three-node triangular and four-node quadrilateral shell elements. A total of 812 eight-node hexahedral, six-node pentahedral, and four-node tetrahedral solid elements were created to mesh three ossicles, two joints (incudomalleolar and incudostapedial joints), and six ligaments/tendons (superior mallear ligament C1, lateral mallear ligament C2, posterior incudal ligament C3, anterior mallear ligament C4, posterior stapedial tendon C5, and tensor tympani tendon C7). Lying in the stapes footplate plane, 25 spring elements were used to model the stapedius annular ligament. A total of 49 spring–dashpot elements perpendicular to the footplate plane were incorporated to model the cochlear fluid (cochlear impedance). The total number of nodes was 1,497, and the number of degrees of freedom was 4,491 in the model shown in Figure 2.36(f). The size of the model was adequate for an FE analysis with a reasonable computation time.

Boundaries of the FE model included suspensory ligaments, intra-aural muscle tendons, tympanic annulus, stapedius annular ligament, and cochlear fluid. As shown in Figure 2.36(f), the malleus, incus, and stapes are attached to the eardrum by the malleus and at the oval window by the stapes footplate. Suspensory ligaments and intra-aural muscle tendons also support the ossicles. Four major suspensory ligaments (superior malleus C1, lateral malleus C2, posterior incus C3, and anterior malleus C4) and two intra-aural muscle tendons (posterior stapedial C5 and tensor tympani C7) were regarded as elastic constraints. The tympanic annulus was modeled as an elastic ring that connects the periphery of the eardrum and the bony wall of the ear canal. The cochlear fluid was assumed as a viscoelastic constraint, C6.

Frequency response analyses were carried out using ANSYS. Excellent resolution of the stapes footplate displacements was observed in six human temporal bones using laser Doppler interferometry over the acoustic frequency ranges, including 80, 90, and 100 dB SPL of acoustic input at the tympanic membrane. The results from the finite element model show that the 3D model of the human middle ear is suitable for further investigation of middle ear mechanics and transfer functions.

7.6.2 TUTORIAL EXAMPLES

The cantilever beam and the thin-walled tube examples discussed in Section 7.4 are included in the tutorial lessons. This beam model was simply prepared for an easy start with both SolidWorks Simulation and Pro/MECHANICA Structure. Default options and values were mostly used in this example. We discuss reducing the 3D beam to a 1D model in Simulation. Note that no such conversion capability is yet available in Pro/MECHANICA. Once the reader is more familiar with one of these two software tools, he or she may move to the second tutorial example, the thin-walled tube, and other examples in later chapters of this book.

7.6.2.1 Cantilever Beam

A cantilever beam of 1 in. × 1 in. × 10 in. was modeled in both Simulation and Pro/MECHANICA. As discussed in Section 7.4.6, Pro/MECHANICA created 12 tetra-elements and employed a p-version solver to create results. Instead of using the default unit system in.-lb_m-sec, a more commonly employed system, in.-lb_f-sec, was chosen. The force added at the front top edge was 1000 lb_f; the

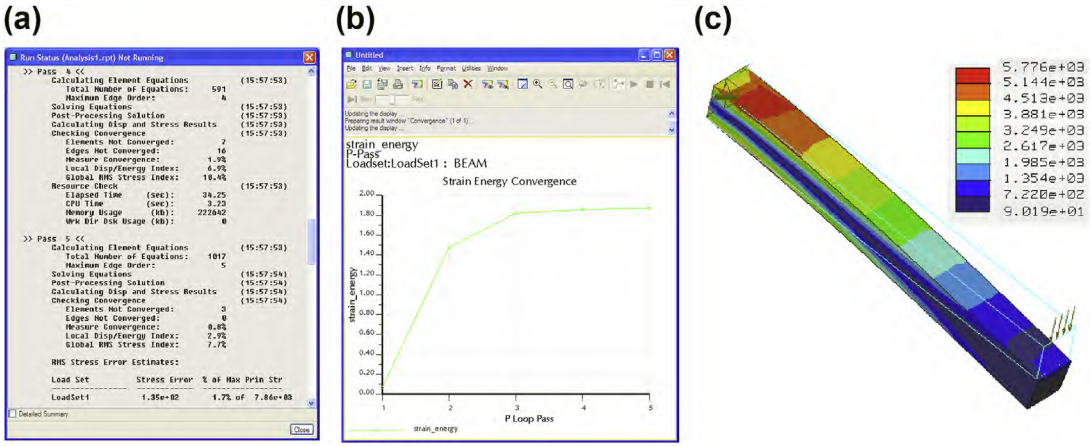


FIGURE 7.39

Simple beam modeled with Pro/MECHANICA: (a) the Run Status window, (b) the strain energy convergence graph, and (c) the von Mises stress fringe plot.

boundary conditions (restrained at the rear end face) and material properties (with modulus $E = 1.06 \times 10^7$ psi and Poisson's ratio $\nu = 0.33$) remained the same. A multipass analysis with 1% strain energy was defined as the convergence criterion. The analysis took five passes (with maximum p-order = 5) and solved 1,017 equations to achieve a 0.8% convergence in strain energy. Essential solution-related information can be found in the Run Status window shown in Figure 7.39(a). The convergence graph shown in Figure 7.39(b) indicates that an excellent convergence was achieved in five passes. The maximum von Mises stress was 5,776 psi at the constrained end, as shown in Figure 7.39(c). Note that the maximum bending stress and vertical displacement were 7,353 psi and 0.03742 in., respectively (not shown in the figure). As before, the displacement result can be verified using classical beam theory. The bending stress would be close to 6,000 psi if Poisson's ratio were set to 0.

The same example was analyzed in SolidWorks Simulation. About 7,500 tetra-elements, as shown in Figure 7.40(a), were created using the default mesh setting, where the global size of the element was 0.2155 in. There were about 36,000 DOF in this model—about 35 times more than in the Pro/MECHANICA model. The maximum vertical displacement was 0.0382 in. (Figure 7.40(b)), which was very close to that of Pro/MECHANICA (0.03742 in.) and the analytical solution. The maximum bending stress shown in Figure 7.40(c) was 6,088 psi (with Poisson's ratio set to 0), which was also close to the analytical solution.

The 3D beam was idealized to a 1D finite element model in Simulation, as shown in Figure 7.41(a). The same load and boundary conditions were applied at respective ends where joints were created when the solid beam was converted to a 1D beam. The auto mesh generator created 46 beam elements (Figure 7.41(b)) with 276 DOF (as opposed to the 36,000 employed in the 3D model of Simulation). Note that for this simple example, one beam element gave exact solutions since the exact solution of the beam displacement was a cubic function and cubic shape functions were employed in element formulation. The bending stress and maximum vertical displacement were 6,000 psi (Figure 7.41(c)) and 0.03778 in., respectively. These results were excellent. Bending stress was identical to that of the analytical solution.

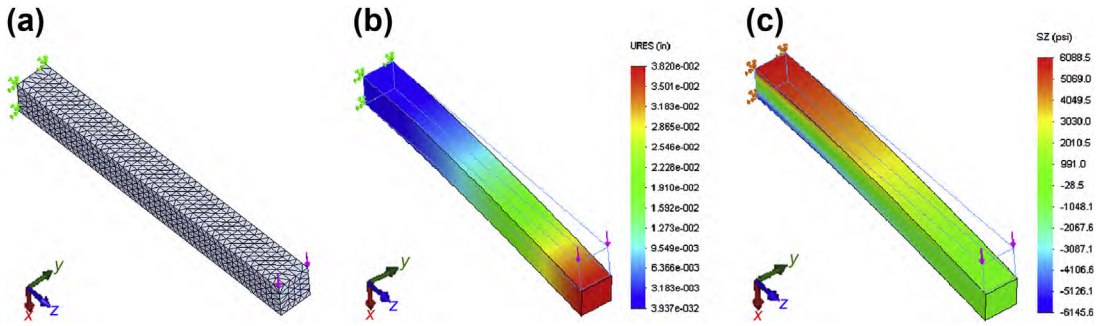


FIGURE 7.40

Simple beam modeled with SolidWorks Simulation: (a) the finite element mesh, (b) the displacement *Ux* (vertical) fringe plot, and (c) the von Mises stress fringe plot.

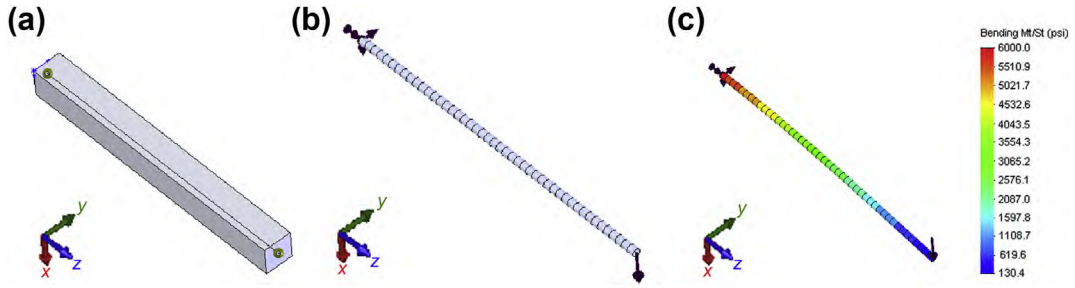


FIGURE 7.41

One-dimensional beam modeled in SolidWorks Simulation: (a) the beam with two end joints, (b) the finite element mesh with load and boundary conditions, and (c) the bending stress fringe plot.

7.6.2.2 Thin-Walled Tube

The second tutorial example was mentioned in Section 7.4.2, where a thin-walled tube was analyzed in both its original 3D form and in half, taking advantage of symmetric conditions as well as a compressed surface model. Both solid and surface models were analyzed in Pro/MECHANICA Structure, as discussed in Section 7.4.2. In this section, we discuss the same models analyzed in SolidWorks Simulation. Step-by-step instructions can be found in the tutorial project S3.

A half-tube model was created in SolidWorks in the SI unit system. The outer radius and height of the large tube (vertical) were 51 and 240 mm, respectively; for the small tube, they were 41 and 100 mm, respectively. The fillet radius was 15 mm, and the thickness was 2 mm over the entire model. The bottom face of the tube was fixed just as in the Pro/MECHANICA model. A symmetry boundary condition was imposed on the faces of the middle section where the model was split in half. A 500-N load was applied at the front face of the small tube, as shown in Figure 7.42(a). The default mesh

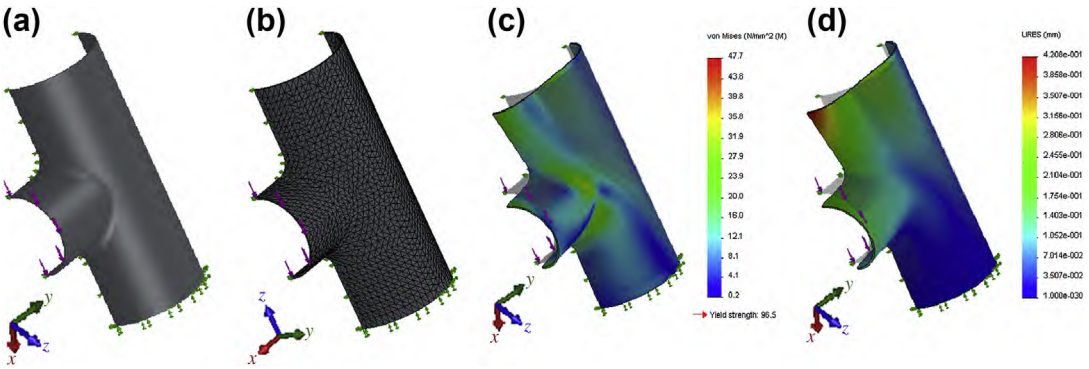


FIGURE 7.42

Thin-walled tubing modeled with SolidWorks Simulation: (a) the solid model with load and boundary conditions, (b) the finite element mesh, (c) the von Mises stress fringe in deformed shape, and (d) the displacement magnitude fringe plot.

setting was chosen (with a global size of 4.383 mm), yielding 14,433 tetra-elements (Figure 7.42(b)). The analysis results shown in Figures 7.42(c) and (d) indicate that the maximum von Mises stress and displacement magnitude were 46.7 MPa and 0.421 mm, respectively—very close to those of Pro/MECHANICA.

An attempt was made to insert mid-surfaces between respective pairs of the outer and inner surfaces to create the surface model, similar to what was done in Pro/MECHANICA Structure. Two mid-surfaces, compressed from the outer and inner surfaces of the large and small tubes, were created successfully. However, SolidWorks failed to create the mid-surface for the fillet because the inner face of the fillet pair was not the exact offset of the outer face.

A surface model, shown in Figure 7.43(a), was then created from scratch. There were about 2,200 triangular elements, as shown in Figure 7.43(b), with about 27,000 DOF using a mesh setting with a

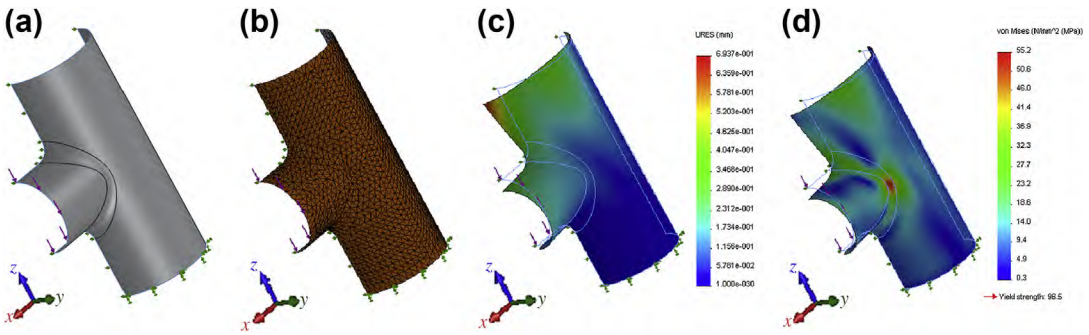


FIGURE 7.43

Tube surface model in SolidWorks Simulation: (a) the surface model with load and boundary conditions, (b) the finite element mesh, (c) the displacement magnitude fringe plot, and (d) the von Mises stress fringe in deformed shape.

global element size of 6.153 mm. Note that this model, in terms of DOF, was about ten times larger than the Pro/MECHANICA model. The maximum displacement was 0.6937 mm (Figure 7.43(c)), which was larger than that of Pro/MECHANICA (0.42 mm). The maximum von Mises stress, shown in Figure 7.43(d), was 55.2 MPa, also larger than that of Pro/MECHANICA (42 MPa). Note that a convergent study may be necessary to verify the accuracy of the FEA results obtained from SolidWorks Simulation. Serious design engineers may use FEA codes, such as ANSYS, MSC/Nastran, or ABAQUS to acquire more dependable FEA results.

7.7 SUMMARY

In this chapter, we discussed methods for structural analysis, both analytical and numerical. Analytical methods often support structural analysis of problems with simple geometry. Usually solutions can be obtained in closed form and solved by hand calculation. However, these methods are not general enough to support problems encountered in engineering design. Realistically, design engineers must depend on FEA for making design decisions that ensure structural integrity and satisfy performance requirements. Although analytical methods are not general, they are indispensable for mechanical engineers to understand the fundamentals in structural analysis and to develop references for verifying FEA results.

We devoted much discussion to finite element methods. We focused our discussion on the essential aspects of using FEA software for modeling and analysis from a practical perspective. We hope this chapter helps readers become more familiar with FEA. Practice should lead to more confidence and competence in using FEA tools for creating adequate models and obtaining reasonable results to support product design.

Although FEA software is powerful, as discussed in this chapter, it is also dangerous. Any mistakes made in modeling, analysis, or even results interpretation can lead to a wrong design decision. FEA users must be very careful in using the software for solving structural problems, learning how to check or verify FEA results before presenting them to others. Again, keep in mind that good engineers make sound judgments; incompetent engineers live with fuzzy guesses.

In this chapter, we focused on the analysis aspect of structural design. Several design technologies for structural optimization, such as shape optimization, will be discussed in Chapter 17 Design Optimization. In the following chapters, we will learn other important aspects of product performance evaluations, including motion analysis in Chapter 8, fatigue fracture analysis in Chapter 9, and reliability analysis in Chapter 10. These chapters should provide readers a good understanding of the essential topics in product performance evaluation.

APPENDIX 7A: THE DEFAULT in.-lb_m-sec UNIT SYSTEM

The default unit system employed by Pro/ENGINEER, and therefore the Pro/MECHANICA Structure is in.-lb_m-sec (inch-pound mass-second). This system is not common to many engineers. The basic physical quantities involved in determining a unit system are length, time, mass, and force. These four basic quantities are related through Newton's second law:

$$F = ma \tag{7A.1}$$

where F, m, and a are force, mass, and acceleration (length per second squared), respectively.

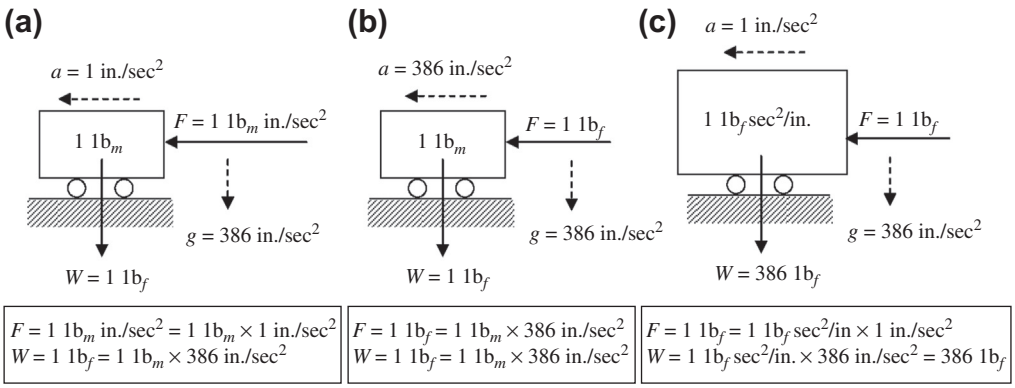


FIGURE 7A.1

(a) A 1 lb_m in./sec^2 force applied to a 1 lb_m mass block, (b) a 1 lb_f force applied to a 1 lb_m mass block, and (c) a 1 lb_f force applied to a 1 lb_f sec^2/in. mass block.

In the default in.-lb_m-sec system, the force unit is determined by length (in.), mass (lb_m), and second (sec) through

$$1 \text{ lb}_m \text{ in.}/\text{sec}^2 (\text{force}) = 1 \text{ lb}_m (\text{mass}) \times 1 \text{ in.}/\text{sec}^2 (\text{acceleration}) \tag{7A.2}$$

where the force unit, lb_m in./sec^2, is a derived unit.

From Eq. 7A.2, a force of 1 lb_m in./sec^2 generates an acceleration of 1 in./sec^2 when applied to a 1 lb_m mass block, as shown in Figure 7A.1(a). The same block weighs 1 lb_f on earth (see Figure 7A.1(b)), where the gravitational acceleration is assumed to be 386 in./sec^2; that is

$$1 \text{ lb}_f (\text{force}) = 1 \text{ lb}_m (\text{mass}) \times 386 \text{ in.}/\text{sec}^2 (\text{acceleration}) \tag{7A.3}$$

Therefore, from Eq. 7A.2 and 7A.3, we have 1 lb_f = 386 lb_m in./sec^2. In other words, the force quantity entered into Pro/MECHANICA Structure is in lb_m in./sec^2 by default, which is 386 times smaller than the 1 lb_f that we are more used to. When a 1 lb_f force is applied to the same mass block, it accelerates 386 in./sec^2, as shown in Figure 7A.1(c). Therefore, care is essential in entering numerical figures when defining analysis models. For example, if a 1,000 unit force is applied to a mechanical component in the default unit system, it is in fact 1,000/386 = 2.59 lb_f, which is very small.

On the other hand, the mass unit, 1 lb_m = 1/386 lb_f sec^2/in., means that a 1 lb_m mass block is 386 times smaller than a 1 lb_f sec^2/in. Therefore, a 1 lb_f sec^2/in. block weighs 386 lb_f on earth. A 1 lb_f force applied to the mass block accelerates at a 1 in./sec^2 rate, as illustrated in Figure 7A.1(c).

QUESTIONS AND EXERCISES

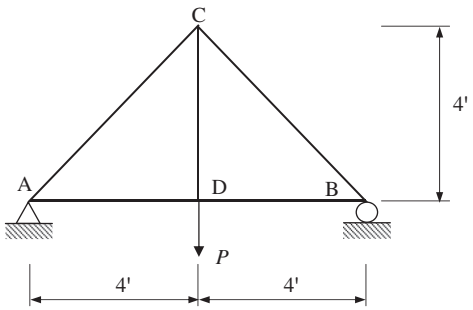
7.1. Calculate the maximum stress and its location for the inclined cantilever beam in Example 7.1 using an analytical method.

7.2. Calculate the rotation angle at the tip of the inclined cantilever beam in Example 7.1 using an analytical method.

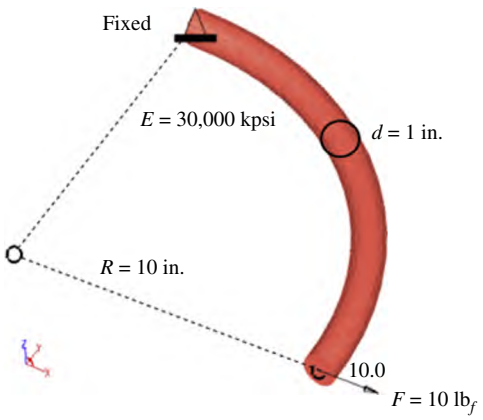
7.3. Calculate the displacement functions of the cantilever beam example in Example 7.4 using Eqs 7.3 and 7.5.

7.4. Repeat Example 7.4 using the following stress functions: $\sigma_x = C_1x^2y$ and $\sigma_x = C_1x^2$. Do any of the two functions provide correct stress results? Is one better than the other?

7.5. The simple pin-connected bars in the figure below are loaded with a force P. If all bars are of equal rigidity *EA*, what are the horizontal and vertical displacements at Point D? Hint: use Castigliano's theorem.



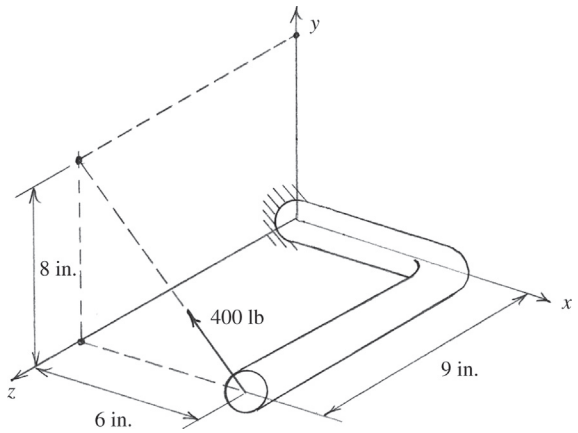
7.6. Create an FEA model to calculate the maximum displacement and stress for the curve beam shown in Example 7.3. The dimensions of the beam are $R = 10$ in. and $d = 1$ in.; the material is steel; $E = 30 \times 10^6$ psi; $F = 10$ lb (see figure below).



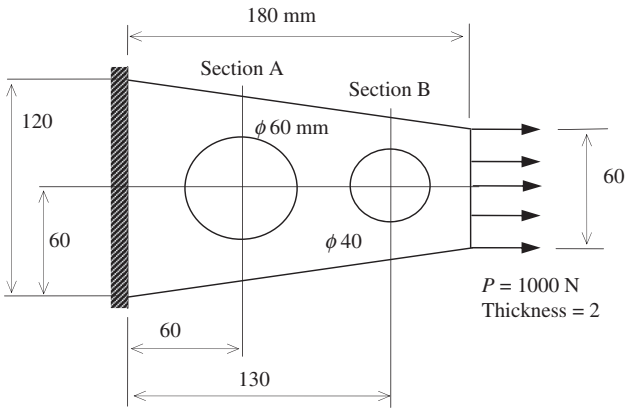
7.7. An L-shaped circular bar of diameter 1.25 in. is loaded with an evenly distributed force of 400 lbs at its end face (see figure on the next page). Note that the elbow radius (the corner of the L-shape) is 1 in. and the material is 1060 steel.

a. Create an FEA model using FEA software to calculate the maximum principal stress and the maximum shear stress in the bar.

b. Calculate the maximum principal stress and maximum shear stress using analytical beam theory. Where are the maximum stresses located? Compare your calculations with those obtained from FEA, both values and locations. Are they close? Why or why not? Comment on your comparison.



7.8. Create and analyze stress for a thin-shelled model of thickness 2 mm (see figure below) using FEA. Assume AISI 1060 steel as the material. Refine the mesh around high-stress areas, and show a screen capture for the mesh and axial stress. Calculate the maximum stresses at both Sections A and B using the stress concentration theory that you learned in a course on strength of materials or design of mechanical components. Check your calculations with those obtained from FEA. Do they agree? Why or why not? Comment on your conclusion.



7.9. **Mini-Project:** The figures that follow show a tracked-vehicle roadarm that connects the roadwheel to the torsion bar of the chassis. The geometric shape of the roadarm is shown in the figure and its cross-sectional dimensions are given in the table. Displacement constraints are

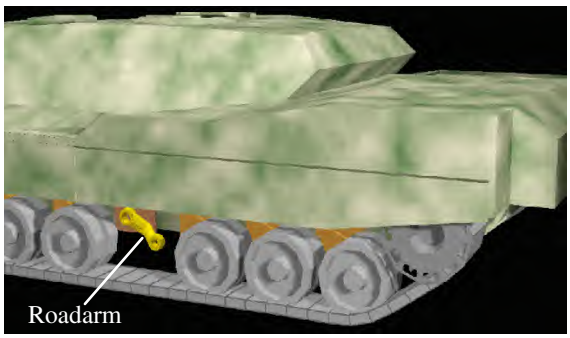
defined at the inner surface of the hole where the torsion bar is attached. A point load is applied at a point that simulates the shaft of the roadwheel, as shown in the bottom figure of next page. Note that the force consists of two components: $F_{x3} = 0.8F$ and $F_{x2} = -0.6F$. The roadarm is made of AISI 1030 HR steel, with material properties of Young's modulus $E = 3.0 \times 10^7$ psi and Poisson's ratio $\nu = 0.3$.

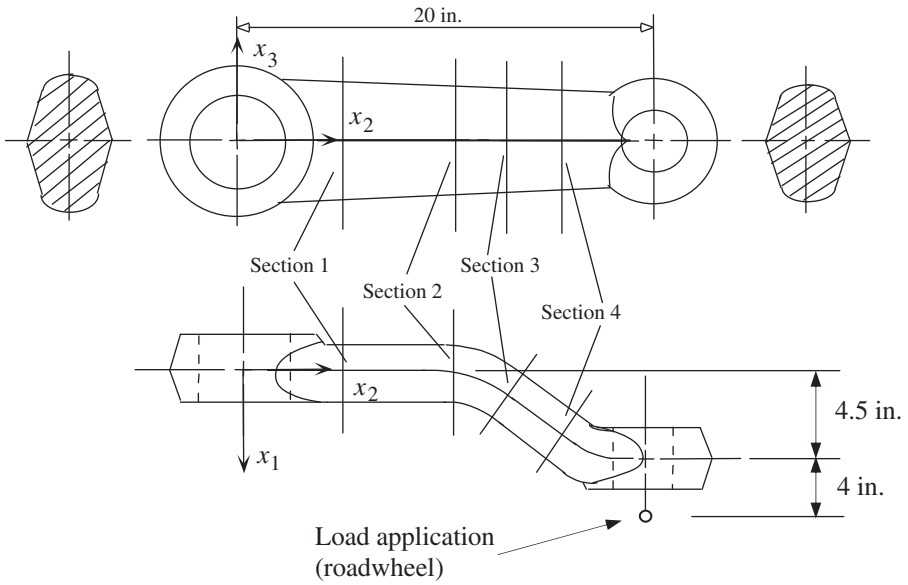
a. If the safety factor is 2, calculate the allowable force F using distortion energy theory.

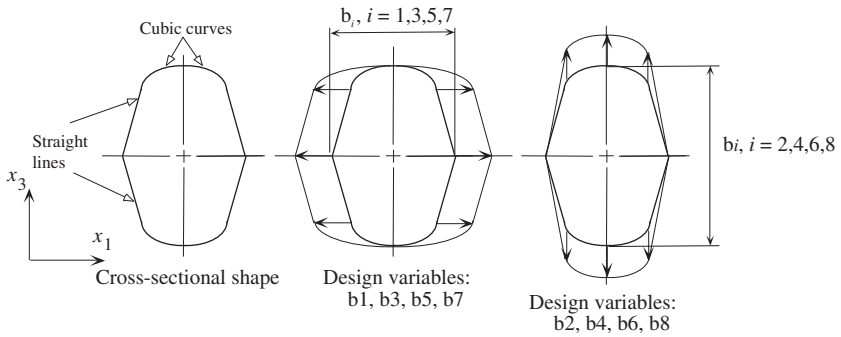
b. Locate the hotspot where maximum stress occurs. Draw a stress element at the spot identified and show all stress components.

c. If the actual force applied to the roadarm is greater than F as calculated in (a), suggest a design change from the given eight cross-sectional dimensions that will reduce the highest stress most effectively.

Note that you will have to make assumptions in order to simplify the problem for a possible solution. You will need to write down all of your assumptions and justify them with reasoning based on engineering judgment. In (a) and (b), you will need to give a quantitative estimate on the possible error in your calculations resulting from these assumptions.







Geometric dimension	Description	Initial design
b1	Width of cross-section 1	1.968 in.
b2	Height of cross-section 1	3.250 in.
b3	Width of cross-section 2	1.968 in.
b4	Height of cross-section 2	3.170 in.
b5	Width of cross-section 3	2.635 in.
b6	Height of cross-section 3	3.170 in.
b7	Width of cross-section 4	2.635 in.
b8	Height of cross-section 4	3.170 in.

REFERENCES

Atkinson, K.E., 1989. An Introduction to Numerical Analysis, second ed. John Wiley.

Babuška, I., Griebel, M., Pitkaranta, J., 1989. Problem of selecting the shape functions for a p-Type finite element. International Journal for Numerical Methods in Engineering 28 (8), 1891–1908.

Bathe, K.J., 2007. Finite Element Procedures. Prentice Hall, 1996.

Beer, F.P., Johnston, E.R., DeWolf, J.T., 2002. Mechanics of Materials, third ed. McGraw-Hill.

Belytschko, T., Chen, J.S., 2007. Meshfree and Particle Methods. John Wiley.

Belytschko, T., Lu, Y.Y., Gu, L., 1994. Element free Galerkin methods. International Journal for Numerical Methods in Engineering 37, 229–256.

Bendsoe, M.P., Sigmund, O., 2003. Topology Optimization, Theory, Methods, and Applications. Springer-Verlag.

Benzley, S.E., Perry, E., Merkley, K., Clark, B., 1995. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In: Proceedings, 4th International Meshing Roundtable. Sandia National Laboratories.

Brauer, J.R. (Ed.), 1993. What Every Engineer Should Know About Finite Element Analysis. Marcel Dekker.

Chang, K.H., Edke, M., 2010. Manufacturing in shape optimization of structural components. In: Linton, R.F., Carroll Jr., T.B. (Eds.), Computational Optimization: New Research Developments. Nova Science Publishers.

Chang, K.H., Magdum, S., Khera, S., Goel, V.K., 2003. An advanced computer modeling and prototyping method for human tooth mechanics study. Annals of Biomedical Engineering 31 (5), 621–631.

Chen, J.S., Pan, C., Wu, T.C., 1997. Large deformation analysis of rubber based on a reproducing kernel particle method. Computational Mechanics 19, 53–168.

Choi, S.-K., Grandhi, R., Canfield, R.A., 2007. Reliability-Based Structural Design. Springer.
Edke, M., Chang, K.H., 2010. Shape Sensitivity Analysis For 2-D Mixed Mode Fractures Using Extended FEM (XFEM) And Level Set Method (LSM). Mechanics Based Design of Structures and Machines 38 (03), 328–347.
Finney, R.H., 2001. Finite element analysis. In: Gent, A.N. (Ed.), Elasticity in Engineering with Rubber: How to Design Rubber Components, second ed. Alan N. Gent. Hanser Verlag.
Grindeanu, I., Chang, K.H., Choi, K.K., Chen, J.S., 1998. Design sensitivity analysis of hyperelastic structures using a meshless method. AIAA Journal 36 (4), 618–627.
Hamrock, B.J., Schmid, B., Jacobson, O., 2005. Fundamentals of Machine Elements, second ed. McGraw-Hill.
Hwang, H.Y., Choi, K.K., Chang, K.H., 1997. Second-order shape design sensitivity analysis using a p-version finite element tool. Journal of Structural Optimization 14, 91–99.
Juvinall, R.C., Marshek, K.M., 2000. Fundamentals of Machine Component Design, third ed. John Wiley.
Li, S., Liu, W.K., 2004. Meshfree Particle Methods. Springer Verlag.
Middleton, J., Jones, M.L., Wilson, A.N., 1990. Three-dimensional analysis of orthodontic tooth movement. Journal of Biomedical Engineering 12, 319–327.
Moës, N., Dolbow, J., Belytschko, T., 1999. A finite element method for crack growth without remeshing. International Journal for Numerical Methods in Engineering 46 (1), 131–150.
Monaghan, J.J., 1988. An introduction to SPH. Computer Physics Communications 48, 88–96.
Nayroles, B., Touzot, G., Villon, P., 1992. Generalizing the finite element method: diffuse approximation and diffuse elements. Computational Mechanics 10, 301–318.
Norton, R.L., 2011. Machine Design: An Integrated Approach, fourth ed. Prentice Hall.
Pilkey, W.D., Wunderlich, W., 2002. Mechanics of Structures: Variational and Computational Methods, second ed. CRC Press.
Quadros, W.R., Ramaswami, K., Prinz, F.B., Gurumoorthy, B., 09/2004. LayTracks: a new approach to automated geometry adaptive quadrilateral mesh generation using medial axis transform. International Journal for Numerical Methods in Engineering 61 (2), 209–237.
Stokes, H., 1995. Solid modeling and the initial graphics exchange specification (IGES). In: LaCourse, D.E. (Ed.), Handbook of Solid Modeling. McGraw-Hill.
Strang, G., Fix, G., 1973. An Analysis of the Finite Element Method. Prentice Hall.
Sun, Q., Chang, K.H., Dormer, K., Dyer, R., Gan, R.Z., 2002. An advanced computer-aided geometric modeling and fabrication method for the human middle ear. Medical Engineering and Physics 24 (9), 595–606.
Szabó, B., Babuska, I., 1991. Finite Element Analysis. Wiley-Interscience.
Tang, P.-S., Chang, K.H., 2001. Integration of Topology and Shape Optimizations for Design of Structural components. Journal of Structural Optimization 22 (1), 65–82.
Wilson, E.L., 2000. Three Dimensional Static and Dynamic Analysis of Structures: A Physical Approach with Emphasis on Earthquake Engineering. Computers and Structures, Inc.
Zienkiewicz, O.C., Taylor, R.L., 1989. The Finite Element Method. McGraw-Hill.

CHAPTER

MOTION ANALYSIS

8

CHAPTER OUTLINE

8.1 Introduction 393
8.2 Analytical Methods 396
8.2.1 Particle Motion 397
8.2.2 Rigid-Body Motion 404
8.2.3 Multibody Kinematic Analysis 407
8.2.4 Multibody Dynamic Analysis 411
8.3 Computer-Aided Methods 415
8.3.1 Kinematic Analysis 415
8.3.2 Kinematic Joints 421
8.3.3 Multibody Dynamic Analysis 424
8.4 Motion Simulation 427
8.4.1 Creating Motion Models 428
8.4.1.1 Ground Parts (or Ground Bodies) 428
8.4.1.2 Moving Parts (or Moving Bodies) 428
8.4.1.3 Constraints 428
8.4.1.4 Degrees of Freedom 430
8.4.1.5 Forces 431
8.4.1.6 Initial Conditions 432
8.4.1.7 Motion Drivers 432
8.4.2 Motion Analysis 432
8.4.3 Results Visualization 433
8.5 Motion Simulation Software 434
8.5.1 General-Purpose Codes 434
8.5.2 Specialized Codes 434
8.6 Case Studies 435
8.6.1 Formula SAE Racecar 435
8.6.2 High-Mobility Multipurpose Wheeled Vehicle 445
8.6.3 Driving Simulators 450
8.6.4 Recreational Waterslides 454

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00008-9
Copyright © 2015 Elsevier Inc. All rights reserved.

8.7 Tutorial Examples **457**
8.7.1 Sliding Block 458
8.7.2 Single-Piston Engine 459
8.8 Summary **461**
Questions and Exercises **461**
References **462**

Many products or mechanical systems involve moving parts. Parts or components must move in certain ways to perform required functionality and achieve desired performance. Components must not collide or interfere with each other during normal operations. Also, the system must be efficient (usually lightweight) yet durable. Essentially, the product must be well designed, and designers must understand the kinematic and dynamic behavior of the system and be sure beforehand that the components of the mechanical system move according to the design intent. Discovering problems after the product is manufactured means it is usually too late and too costly to correct them. Motion analysis offers viable alternatives to support designers in simulating and analyzing the movement of parts in mechanical systems in the early design stages.

Traditionally, motion analysis has been divided into two major categories: kinematics and dynamics. Kinematics is the study of motion without regard to the forces that cause it. Dynamics, on the other hand, is the study of motion that results from forces. Except for simple ones, such as particle dynamics, very few of these problems can be solved by hand using analytical equations. The majority must be solved using numerical methods. Also, most problems must be formulated in a certain way so that they can be generalized and handled by computer software—that is, so-called computer-aided kinematic and dynamic analyses. Although computer-aided approaches and software tools are powerful, understanding analytical methods is crucial. The underlying physics helps us understand how mechanisms behave in certain ways in a given motion scenario. Understanding the underlying physics also helps us make good modeling decisions and choose adequate simulation parameters that lead to faster and more reliable simulations.

Furthermore, it is critical that we verify numerical results obtained from computer tools whenever possible. Very often, analytical solutions for complex problems are not available. However, it is good practice, especially for new users, to start with something simple so that the solutions obtained from the motion analysis software can be verified analytically. The bottom line is that we must have adequate knowledge to use the computer tools correctly and effectively.

In addition to analytical methods, we devote most of the discussion in this chapter to computer-aided approaches for kinematic and dynamic analyses. Note that our focus is not on computational theory but rather on the essential elements for motion modeling and analysis from a practical perspective. The goal of this chapter is to help the reader become confident and competent in using motion software tools for creating adequate models and obtaining reasonable results to support design. Those who are interested in kinematics and dynamics theory can refer to other excellent books, such as *Computer Aided Kinematics and Dynamics of Mechanical Systems*, by Haug, or *Dynamics: Theory and Applications*, by Kane and Levinson. (Both are listed in the references at the end of the chapter.) Major motion simulation software packages, both general-purpose and specialized, are presented to provide a general understanding of tool availability and sufficient information to make adequate choices.

Several practical examples are introduced. Two vehicle examples—a Formula SAE racecar and a high-mobility multipurpose wheeled vehicle (HMMWV)—illustrate vehicle suspension simulation and design. Driving simulators that employ real-time simulation for vehicle dynamics illustrate some of the most advanced motion simulation applications. Also, a theme park waterslide application shows the variety of engineering problems that motion analysis supports. Finally, two tutorial examples deal respectively with a sliding block and a single-piston engine using both Pro/ENGINEER Mechanism Design and SolidWorks® Motion. These practice examples can also be found in the tutorial lessons. Detailed instructions for bringing up these models and the steps for carrying out motion simulations can be found in Projects P2 and S2 of this book. Example models are available for download at the book's companion website (http://booksite.elsevier.com/9780123820389).

Overall, the objectives of this chapter are to provide basic kinematics and dynamics theory using simple examples to further understanding of the underlying computation methods for motion analysis; to familiarize readers with motion modeling and analysis for effective use of motion analysis tools to support product design; to familiarize readers with existing commercial motion analysis software; to instruct readers in the use of either Pro/ENGINEER Mechanism Design or SolidWorks Motion for basic applications, with the help of the tutorial lessons.

8.1 INTRODUCTION

Motion analysis comprises the physical laws and mathematics required to study and predict the performance and behavior of mechanical systems. When a product design is first created in a CAD environment, it is unclear if the system will behave as intended, if the components will move in the designated manner, or collide or interfere. Also uncertain is the system's functionality and overall performance. Certainly these questions can be answered by a functional prototype of the product. However, since it is usually too late and too costly to correct design problems by then, it is crucial that design engineers understand how the system behaves and are able to predict if the design meets the functional requirements. Instead of building a functional prototype, motion analysis offers a viable alternative to answer some of these design questions with accuracy that is adequate to support design decision making. In practice, motion analysis supports engineering design by simulating the kinematic and dynamic performance of the product, and it proves the soundness of a design without dependence on physical testing, especially in early design stage.

A broad range of products and mechanical systems involve moving parts. Some of these products, as shown in Figure 8.1, include mechatronics devices that we use on a daily basis such as computer hard disk drives, heavy construction equipment (e.g., backhoes), ground vehicles (e.g., the Army's high-mobility multipurpose wheeled vehicle), and theme park rides such as a waterslide. Design objectives and functional requirements vary for different products. For a hard disk drive, increasing the spindle speed, for example, from 7,200 rpm to 10,000 rpm, significantly increases the data-accessing speed in our computers. For a backhoe, power and torque must be sufficient to generate adequate digging forces while minimizing the impact force and the mechanical vibration felt by the operator. In the case of the HMMWV, maneuverability is critical since the vehicle is intended to be driven on rough roads. As for the waterslides, safety is always number one. Besides functionality, structural integrity must be ensured. Reaction forces between moving parts may cause part failures due to yield, buckle, or fatigue.

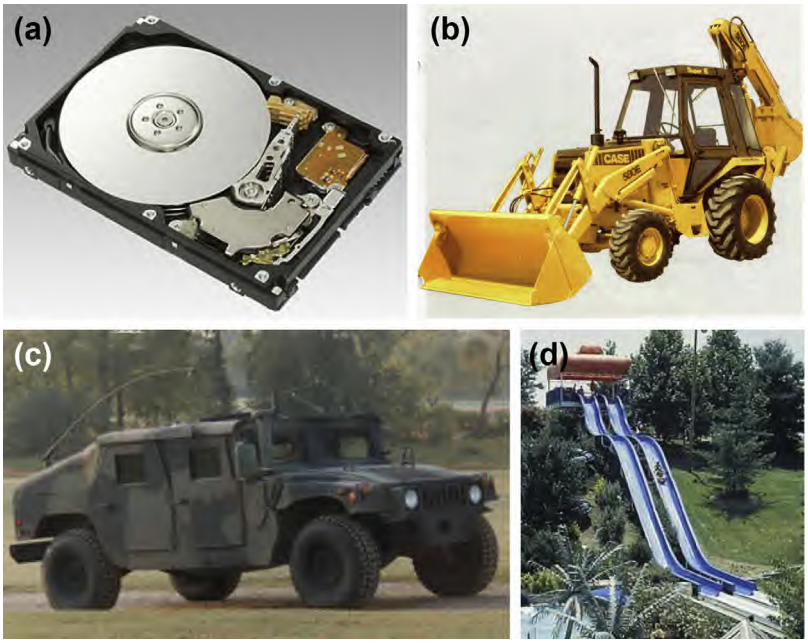


FIGURE 8.1

Complex mechanical systems: (a) mechatronic device—hard disk drive (source: http://upload.wikimedia.org/wikipedia/commons/1/1e/Hard_disk_dismantled.jpg); (b) heavy equipment—backhoe (source: http://www.docstoc.com/docs/85268831/Transportation-backhoe); (c) ground vehicle—HMMWV (source: http://upload.wikimedia.org/wikipedia/commons/7/72/Iraqi_Humvees.jpg); (d) theme park ride—waterslide.

Most products involve multiple moving parts. From a motion analysis perspective, such products are often referred to as multibody systems, which have to be modeled or formulated by incorporating mass and inertia properties of individual parts as well as the connections (also called joints) between parts. In most cases, these parts are assumed rigid and so are referred to as rigid-body systems. In some cases, certain parts have to be considered flexible, such as the arm of the read/write head in a hard drive, which deforms substantially during motion. The arm that carries the read/write head flies over the surface of the disk and must reach precise locations in a very short time. The arm deforms mainly as a result of inertia. Position and velocity of the read/write head must be predicted in high precision, considering the deformation. However, not all motion problems are complex. Some applications can be simplified to particle dynamics. For example, the rider on the waterslide can be assumed a particle with concentrated mass when the engineer is carrying out motion simulation for safety analysis. Certainly for a more detailed simulation the rider itself must be modeled as a multibody system.

The complexity of a motion simulation model varies. An HMMWV suspension model can have as little as 14 bodies. A body is an entity in a motion analysis model and can be stationary (as a ground body) or in motion; however, no relative motion is allowed between parts within a body. The number of

bodies in a mechanical system can rise easily, which increases the size of the motion model and therefore requires more computational time to solve the problem. A motion model can be simple and also very complex, depending on the level of fidelity retained in it. Determination of the level of model fidelity is often driven by the questions we want the motion simulation to answer.

Some questions require knowledge of position, velocity, and acceleration of individual parts in the system, and they can often be answered by conducting a kinematic analysis. Kinematic analysis provides the physical positions, velocity, and acceleration of all parts in an assembly with respect to time, without consideration of the forces leading to the motion. This analysis is also useful for evaluating motion for interference of complex mechanical systems. On the other hand, questions involving forces must be answered by dynamic analysis, which is the study of motion in response to externally applied loads. This analysis is necessary when torque or force is involved in motion analysis—for example, the torque required to generate sufficient digging force for a backhoe. In those cases, the dynamic behavior of a mechanism is governed by Newton's laws of motion.

Even though design questions vary and they often must be determined for specific applications of certain fidelity levels, there are a few basic questions that are common and typical for all motion applications. A single-piston engine, shown in Figure 8.2, is used to illustrate some typical questions:

- Will the components of the mechanism collide or interfere in operation? For example, will the connecting rod collide with the inner surface of the piston or the inner surface of the engine case during operation?
- Will the components in the mechanism move according to our intent? For example, will the piston stay entirely in the piston sleeve? Will the system lock up when the firing force aligns vertically with the connecting rod?
- How much torque or force does it take to drive the mechanism? For example, what is the minimum firing load to move the piston? Note that in this case, proper friction forces must be added to simulate the resistance between moving parts before a realistic firing force can be calculated.
- How fast will the components move (e.g., what is the linear velocity of the piston)?
- What is the reaction force or torque generated at a connection between components (or bodies) during motion? For example, what is the reaction force at the joint between the connecting rod and the piston pin? This reaction force is critical since the structural integrity of the piston pin and the connecting rod must be ensured—that is, they must be strong and durable enough to sustain the load in operation.

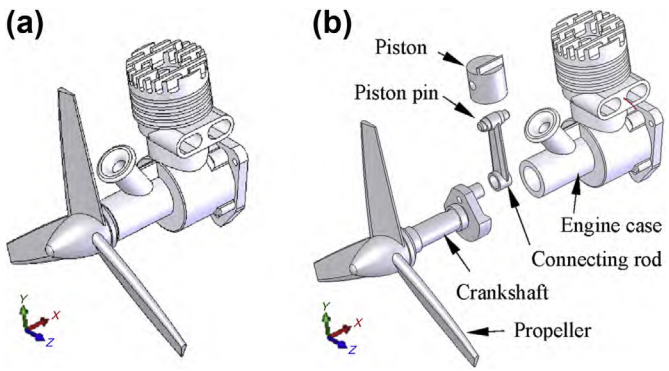


FIGURE 8.2

A single-piston engine: (a) unexploded view and (b) exploded view.

Motion simulation helps us answer these common questions accurately and realistically, as long as the motion model is properly created. Some motion analysis tools also help us search for better design alternatives. A better design alternative is very much problem-dependent. It is critical that a design problem be clearly defined by the designer up front before searching for better design alternatives. For the engine example, a better alternative can be a design that meets the functional requirements and reveals, for example,

- A smaller reaction force applied to the connecting rod
- No collisions or interference between components.

A common approach to searching for design alternatives is to vary the component sizes of the mechanical system. Vary component sizes to explore better design alternatives, the parts and assembly must be properly parameterized to capture design intents. At the parts level, design parameterization implies creating solid features and relating dimensions so that when a dimension value is changed the part can be properly rebuilt. At the assembly level, design parameterization involves defining assembly mates and relating dimensions across parts. When an assembly is fully parameterized, a change in dimension value can be automatically propagated to all parts affected. Parts affected must be rebuilt successfully; at the same time, they must maintain proper position and orientation with respect to one another without violating any assembly mates or revealing part penetration or excessive gaps. For example, in the single-piston engine (see Chapter 5, Figure 5.1), a change in the bore diameter of the engine case alters not only the geometry of the engine case itself but also all other associated parts, such as the piston, the piston sleeve, and even the crankshaft, as illustrated. Moreover, all parts have to be properly rebuilt and the entire assembly must stay intact through assembly mates.

In this chapter, we start by briefly reviewing analytical methods and then computer-aided approaches for kinematic and dynamic analyses. Essential elements in using computer-aided software for modeling and analysis of motion problems, such as joints, force elements, and the like, are discussed in detail. We touch a bit on the topic of real-time simulation, which realizes the vision of operator-in-the-loop simulation and which has led to several driving simulators. We also discuss commercial motion software, both general-purpose and specialized. Example problems modeled and solved using some of the commercial codes are presented as well.

8.2 ANALYTICAL METHODS

Analytical methods employ analytical formulations to solve motion problems. They are applicable to simple problems, such as particle dynamics, kinematics, and single-body dynamics. Even though the formulations are based on physical laws and principles, they usually have to be derived and solved for individual problems. The formulations lead to equations of motion that are second-order differential equations, sometimes coupled. These differential equations are often solved numerically. Closed-form solutions are available only for some simpler problems. Assumptions are always made up front in formulating or deriving equations of motion. For example, a moving object must be assumed to be of concentrated mass for particle dynamic problem.

As with any assumptions in engineering that help simplify problems, the more the assumptions of a motion model deviate from the physical problem, the less useful the analysis result will be. In general, analytical solutions cannot directly support product design. However, they provide references that support complex motion problem verification.

This section briefly reviews analytical methods for particle dynamics, kinematics of mechanisms, and rigid-body dynamics. It is not intended to be an in-depth presentation and thorough discussion of the subject. The purpose is just for the reader to understand the basics that support problem solving using software tools. Both Newton's law and the energy method are discussed for simple examples. Analytical methods can only go so far. General multibody dynamics analysis must rely on computer-aided methods and computer software for solutions.

8.2.1 PARTICLE MOTION

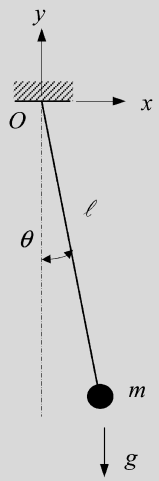
Particle motion is the simplest motion problem to solve. In general, moving objects are assumed to be particles with concentrated mass. A particle moves in only three directions (three translational degrees of freedom). No rotation is considered.

There are two main approaches for deriving equations of motion for particle motion problems: Newton's law and the energy method. Newton's law is straightforward; however, it is limited to very simple problems. A free-body diagram is usually drawn for the particle in motion; then Newton's second law, $F = ma$, is employed to create an equation of motion as a differential equation. In terms of the energy method, kinetic energy and potential energy of the moving particle are first stated. Either the energy conservation law or the more advanced Lagrange equations of motion can be employed to derive differential equations that govern particle motion.

A simple pendulum example is discussed next to show how the equations of motion can be derived using Newton's method, and how the differential equations can be solved analytically.

EXAMPLE 8.1

A simple pendulum shown in the figure is released from a position slightly off the vertical line. Once released, it rotates freely due to gravity. Calculate the position, velocity, and acceleration of the pendulum.



Continued

EXAMPLE 8.1—cont'd

Solution

There are four assumptions that we have to make to apply particle dynamics theory to this simple problem:

- The mass of the rod is negligible.
- The sphere is of a concentrated mass.
- The rotation angle is small.
- No friction is present.

From the free-body diagram shown below, the moment equation at the origin (point O) about the z-axis (normal to the paper) can be written as

$$\sum M = -mg\ell \sin\theta = I\ddot{\theta} = m\ell^2\ddot{\theta} \tag{8.1}$$

Thus,

$$\ddot{\theta} + \frac{g}{\ell}\sin\theta = 0 \tag{8.2}$$

and

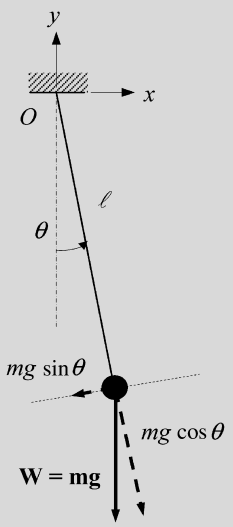
$$\ddot{\theta} + \frac{g}{\ell}\theta = 0 \tag{8.3}$$

when $\theta \approx 0$.

It is well known that the solution of the differential equation is

$$\theta = A_1 \cos\omega_n t + A_2 \sin\omega_n t \tag{8.4}$$

where $\omega_n = \sqrt{\frac{g}{\ell}}$, and A_1 and A_2 are constants to be determined by initial conditions.



The initial conditions for the pendulum are $\theta(0) = \theta_0$, and $\dot{\theta}(0) = 0$. Plugging the initial conditions into the solution, we obtain $A_1 = \theta_0$ and $A_2 = 0$. Therefore, the solutions are

$$\theta = \theta_0 \cos\omega_n t \tag{8.5a}$$

$$\dot{\theta} = -\theta_0\omega_n \sin\omega_n t \tag{8.5b}$$

$$\ddot{\theta} = -\theta_0\omega_n^2 \cos\omega_n t \tag{8.5c}$$

Equations (8.5a) through (8.5c) represent angular position, velocity, and acceleration of the pendulum or, more precisely, the revolute joint at point O that allows the rotation motion of the rod and sphere.

EXAMPLE 8.1—cont'd

Note that the four assumptions stated up front must be incorporated into the motion simulation models if the solutions obtained in the example are to be employed to verify the motion analysis using computer tools. In general, the simulation model and the conditions of the analytical solution must be consistent when the solutions are employed for the verification of the simulation model.

For simple problems, such as the pendulum in Example 8.1, Newton's method works well. However, it is somewhat limited. The energy method is often more powerful in formulating equations of motion for more complex problems. One of the simplest forms of the energy method is the conservation of mechanical energy, which states that the total mechanical energy, which is the sum of the kinetic energy T and the potential energy U, is a constant with respect to time:

$$\frac{d}{dt}(T+U)=0 \tag{8.6}$$

For the simple pendulum example, the kinetic and potential energy are, respectively, $T=\frac{1}{2}I\dot{\theta}^2$, where I is the mass moment of inertia; that is, $I=m\ell^2$ and $U=mg\ell(1-\cos\theta)$. Hence, from Eq. 8.6,

$$\frac{d}{dt}\left(\frac{1}{2}m\ell^2\dot{\theta}^2+mg\ell(1-\cos\theta)\right)=(m\ell^2\ddot{\theta}+mg\ell\sin\theta)\theta=0 \tag{8.7}$$

Therefore,

$$\ddot{\theta}+\frac{g}{\ell}\sin\theta=0 \quad \text{and} \quad \ddot{\theta}+\frac{g}{\ell}\theta=0 \quad \text{where} \quad \theta\approx 0$$

which is identical to Eqs 8.2 and 8.3. Note that the same equation of motion has been derived with two different approaches.

A more general form of the energy method is based on Lagrange's equations. In a dynamic system with n degrees of freedom it is usually possible to choose n geometric quantities, $\boldsymbol{q}=[q_1, q_2,\ldots, q_n]^{\mathrm{T}}$, which uniquely specify the position of all moving components in the system. These quantities are known as generalized coordinates. For example, in the case of the simple pendulum the position of the mass is completely determined by the angle θ. The angle θ is a generalized coordinate of the simple pendulum problem. Lagrange's equation of motion based on Hamilton's principle for particle dynamics (Greenwood, 1987) can be stated as

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\boldsymbol{q}}}\right)-\frac{\partial L}{\partial\boldsymbol{q}}=\boldsymbol{Q} \tag{8.8}$$

where the Lagrangian L is defined as $L\equiv T-U$, $\dot{\boldsymbol{q}}=\partial q/\partial t$. When the system is conservative, $\boldsymbol{Q}=0$. For a nonconservative system (for example, if friction is present), $\boldsymbol{Q}=\boldsymbol{F}$, where $\boldsymbol{F}$ is the vector of generalized forces.

For the simple pendulum example, $\boldsymbol{q}=[\theta]$, $L=T-U=\frac{1}{2}m\ell^2\dot{\theta}^2-mg\ell(1-\cos\theta)$, and $\boldsymbol{Q}=0$. Thus, using Eq. 8.8, we have

$$\frac{d}{dt}(m\ell^2\dot{\theta})-(-mg\ell\sin\theta)=m\ell^2\ddot{\theta}+mg\ell\sin\theta=0 \tag{8.9}$$

Therefore, $\ddot{\theta}+\frac{g}{\ell}\sin\theta=0$, which is the same as Eq. 8.2 (and Eq. 8.3 if $\theta\approx 0$).

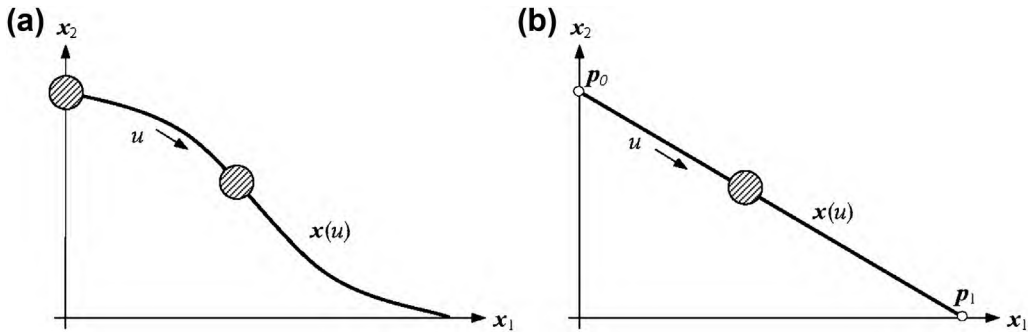


FIGURE 8.3

Object sliding along a curve $\boldsymbol{x}(u)$: (a) general curve and (b) straight line.

Now consider a particle sliding along a slope defined by a parametric curve $\boldsymbol{x}(u)$, shown in Figure 8.3(a), with no friction. Note that the curve can be planar or spatial. The position, velocity, and acceleration of the particle can be obtained by solving the equation of motion derived from the Lagrange equation. The kinetic energy and potential energy are, respectively, $T = \frac{1}{2}m\dot{\boldsymbol{x}}^2 = \frac{1}{2}m(\boldsymbol{x}_{,u}\dot{u})^2$ and $U = mgx_2(u)$, where m is the particle mass and $\boldsymbol{x}_{,u} = \partial\boldsymbol{x}/\partial u$. The Lagrangian is $L = \frac{1}{2}m(\boldsymbol{x}_{,u}\dot{u})^2 - mgx_2(u)$. Using Eq. 8.8, the equation of motion can be derived as

$$\boldsymbol{x}_{,u}^2\ddot{u} + 2\boldsymbol{x}_{,u}\boldsymbol{x}_{,uu}\dot{u}^2 + gx_{2,u} = 0 \tag{8.10}$$

This is an ordinary second-order differential equation of u. Note that the equation of motion shown in Eq. 8.10 can be used in applications such as rollercoasters, where position, velocity, and acceleration of riders modeled as concentrated mass are calculated. Equation 8.10 can be extended to support particle motion on a spatial parametric surface $\boldsymbol{x}(u,w)$ for applications such as waterslides and bobsleds. More on this topic is given in Section 8.6.

Note that, in general, Eq. 8.10 must be solved using a numerical method. Only very simple cases—for example if $\boldsymbol{x}(u)$ is a straight line as shown in Figure 8.3(b)—can be solved analytically. The following example illustrates the details.

EXAMPLE 8.2

A particle slides along a straight line, as shown in Figure 8.3(b), from $\boldsymbol{p}_0$ to $\boldsymbol{p}_1$, where $\boldsymbol{p}_0 = [0,1]$ and $\boldsymbol{p}_1 = [1,0]$. Calculate the position, velocity, and acceleration of the particle and the time required for it to reach point $\boldsymbol{p}_1$. Assume the initial velocity to be zero.

Solution

The parametric equation of a straight line is

$$\boldsymbol{x}(u) = \boldsymbol{p}_0(1-u) + \boldsymbol{p}_1 u = [u, 1-u] \tag{8.11}$$

EXAMPLE 8.2—cont'd

Also,

$$\boldsymbol{x}_{,u}(u) = [1, -1], \boldsymbol{x}_{,uu}(u) = 0, (\boldsymbol{x}_{,u}(u))^2 = \boldsymbol{x}_{,u}(u)\boldsymbol{x}_{,u}(u)^{\mathrm{T}} = 2, \text{ and } x_{2,u} = -1$$

Therefore, the equation of motion becomes

$$\ddot{u} - \frac{1}{2}g = 0 \tag{8.12}$$

We integrate twice to obtain

$$\dot{u} = \frac{1}{2}gt + c_1 \tag{8.13a}$$

and

$$u = \frac{1}{4}gt^2 + c_1 t + c_2 \tag{8.13b}$$

Note that $c_1 = 0$ due to initial velocity, and $c_2 = 0$ due to initial position; that is, $u(0) = 0$, so

$$\boldsymbol{x}(u) = \left[\frac{1}{4}gt^2, 1 - \frac{1}{4}gt^2\right], \quad \dot{\boldsymbol{x}}(u) = \left[\frac{1}{2}gt, -\frac{1}{2}gt\right], \quad \text{and} \quad \ddot{\boldsymbol{x}}(u) = \left[\frac{1}{2}g, -\frac{1}{2}g\right] \tag{8.14}$$

The time required to reach point $\boldsymbol{p}_1$, where $u = 1$, is $t = 2/\sqrt{g}$. Note that all of these results can be verified using Newton's method as well. This is left as an exercise.

For a system of multiple particles, the forces acting on the ith particle, both external and internal, can be written as

$$m_i \ddot{\boldsymbol{r}}_i = \boldsymbol{F}_i + \sum_{j=1}^{n} f_{ij} \tag{8.15}$$

where

m_i is the mass of the ith particle.
$\boldsymbol{r}_i$ is the particle's position vector relative to the fixed reference.
$\boldsymbol{F}_i$ is the force acting on the ith particle.
$\boldsymbol{f}_{ij}$ is the interaction force of the jth particle acting on the ith particle.

In addition to Newton's method, shown in Eq. 8.15, the energy method is applicable and useful for multiparticle problems. For a multiparticle system, the kinetic energy is the sum of the energy of individual particles; so is the potential energy.

EXAMPLE 8.3

This problem is slightly modified from Greenwood (1987). A particle of mass m moves on a frictionless plane. It is connected to a second identical particle by an inextensible string that passes through a small hole at point O, as shown in the figure. The second particle moves only vertically through O. The initial conditions are $r(0) = r_0$, $\dot{r}(0) = 0$, $\dot{\theta}(0) = \omega_0$.

The problem can be divided into three parts: (1) find the minimum initial angular velocity $\omega_{\min}$—that is, when $r(0) = r_0$ is given—so that the second particle is stationary. If the initial angular velocity is $\omega_0 = \frac{1}{\sqrt{3}}\omega_{\min}$, (2) find the maximum and minimum r of the first particle and (3) find the tension force between the particles at both the maximum and minimum r. Note that the maximum and minimum r occur when $\dot{r} = 0$.

Continued

EXAMPLE 8.3—cont'd

Solution

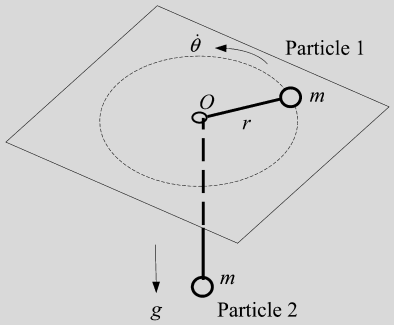
Part 1 is straightforward. The second particle is assumed stationary: $\dot{r} = \ddot{r} = 0$. Hence, from the free-body diagrams shown below, the following can be obtained by applying Eq. 8.15:

$$f = mg = mr\dot{\theta}^2 = mr_0\omega_0^2 \tag{8.16}$$

Thus,

$$\omega_0 = \sqrt{\frac{g}{r_0}} = \omega_{\min} \tag{8.17}$$

That is, if $\omega_0 = \sqrt{\frac{g}{r_0}}$, the second particle is stationary.

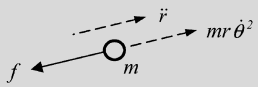


For part 2, if the initial angular velocity is $\omega_0 < \sqrt{\frac{g}{r_0}} = \omega_{\min}$, then $mg > mr_0\omega_0^2$. The first particle starts moving inward toward point O and the second particle moves downward. Since the angular momentum of the system is preserved, the angular velocity of the first particle is

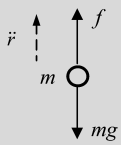
$$H = mr^2\dot{\theta} = mr_0^2\omega_0 \tag{8.18}$$

Hence,

$$\dot{\theta} = \left(\frac{r_0}{r}\right)^2\omega_0 \tag{8.19}$$



Free-body diagram of Particle 1



Free-body diagram of Particle 2

which provides important information about the relation between r and $\dot{\theta}$. The first particle rotates faster when it comes closer to point O. In the meantime, its kinetic energy increases. The first particle moves inward and reaches a minimum r; then it moves outward and reaches the maximum r. The particle continues the movement because no friction is present.

EXAMPLE 8.3—cont'd

The maximum and minimum rotating radii r_{max} and r_{min}, which happen when $\dot{r} = 0$, are to be found. Apparently, Eqs 8.18 and 8.19 are not enough to solve the problem. We need more equations, such as energy conservation. The total mechanical energy of the system is

$$E = T + U = \left(\frac{1}{2}m\dot{r}^2 + \frac{1}{2}mr^2\dot{\theta}^2\right)_1 + \left(\frac{1}{2}m\dot{r}^2 + mgr\right)_2 = m\dot{r}^2 + \frac{1}{2}mr^2\dot{\theta}^2 + mgr \tag{8.20}$$

where the subscripts 1 and 2 represent the energy of the first and second particles, respectively. Also, mgr is the potential energy of the second particle, assuming $U_2 = 0$ when $r = 0$.

From the initial conditions, we have

$$E = \frac{1}{2}mr_0^2\omega_0^2 + mgr_0 \tag{8.21}$$

We bring Eq. 8.19 into Eq. 8.20 and equate it with Eq. 8.21, which yields

$$m\dot{r}^2 + m\frac{r_0^4\omega_0^2}{2r^2} + mgr = \frac{1}{2}mr_0^2\omega_0^2 + mgr_0 \tag{8.22}$$

To find the minimum r, we set $\dot{r} = 0$; therefore, Eq. 8.22 becomes

$$r^3 - \left(r_0 + \frac{r_0^2\omega_0^2}{2g}\right)r^2 + \frac{r_0^4\omega_0^2}{2g} = (r - r_0)\left(r^2 - \frac{r_0^2\omega_0^2}{2g}r - \frac{r_0^3\omega_0^2}{2g}\right) = 0 \tag{8.23}$$

Solving Eq. 8.23, we have $r = r_0$ and

$$r = \frac{r_0^2\omega_0^2}{4g}\left(1 \pm \sqrt{1 + \frac{8g}{\omega_0^2 r_0}}\right) \tag{8.24}$$

Certainly, r cannot be less than zero, so only the positive solution is physically meaningful. If

$$\omega_0 = \sqrt{\frac{g}{3r_0}},$$

the minimum distance is

$$r = \frac{r_0^2\omega_0^2}{4g}\left(1 + \sqrt{1 + \frac{8g}{\omega_0^2 r_0}}\right) = \frac{1}{2}r_0.$$

So in this case, the maximum distance is $r = r_0$.

For part 3 of the problem, we have, from the free-body diagrams, the following two equations:

$$mr\dot{\theta}^2 - f = m\ddot{r} \tag{8.25a}$$

$$f - mg = m\ddot{r} \tag{8.25b}$$

By adding Eqs 8.25a and 8.25b, we have

$$\ddot{r} = \frac{1}{2}r\dot{\theta}^2 - \frac{1}{2}g = \frac{1}{2}r\left[\left(\frac{r_0}{r}\right)^2\omega_0\right]^2 - \frac{1}{2}g \tag{8.26}$$

For $r = r_0$, we have

$$\ddot{r} = \frac{1}{2}r_0\frac{g}{3r_0} - \frac{1}{2}g = -\frac{1}{3}g$$

Continued

EXAMPLE 8.3—cont'd

For $r = 1/2r_0$, we have

$$\ddot{r} = \frac{r_0^3 g}{6r^3} - \frac{1}{2}g = \frac{4}{3}g - \frac{1}{2}g = \frac{5}{6}g$$

Bringing these into Eq. 8.25b, we have

$$f|_{r=r_{\max}} = mg + m\ddot{r} = mg - \frac{1}{3}mg = \frac{2}{3}mg \tag{8.27a}$$

and

$$f|_{r=r_{\min}} = mg + m\ddot{r} = \frac{11}{6}mg \tag{8.27b}$$

8.2.2 RIGID-BODY MOTION

In physics, rigid-body dynamics is the study of the motion of rigid bodies in response to external loads. Unlike particles, which move only in three degrees of freedom (translation in three directions), rigid bodies occupy space and have geometrical properties, such as center of mass, mass moment of inertia, and so forth, that characterize motion in six degrees of freedom (translation in three directions plus rotation in three directions). Rigid bodies are also nondeformable, as opposed to deformable bodies. As such, rigid-body dynamics is used heavily in analyses and computer simulations of physical systems and machinery where rotational motion is important but material deformation does not have a significant effect on the motion of the system.

Two equations are particularly important for the discussion of rigid-body motion using Newton's method. The first is the translation equation of motion given as

$$\boldsymbol{F} = m\ddot{\boldsymbol{x}}_c \tag{8.28}$$

This equation states that the vector sum of the external forces acting on a rigid body is equal to the total mass of the body times the absolute acceleration of the center of mass $\ddot{\boldsymbol{x}}_c$. The second equation is the rotational equation of motion given as

$$\boldsymbol{M} = \dot{\boldsymbol{H}} \tag{8.29}$$

where the reference point for calculating the applied moment $\boldsymbol{M}$ and the angular momentum $\boldsymbol{H}$ is either fixed in an inertial frame or located at the mass center of the rigid body.

If a rigid body rotates about a reference point O, as shown in Figure 8.4, the angular momentum of a small element dV relative to point O is

$$d\boldsymbol{H}_O = \boldsymbol{x} \times dm\dot{\boldsymbol{x}} \tag{8.30}$$

where

$\boldsymbol{x}$ is the position vector of the small element relative to the reference point O.
dm is the mass of the small element dV.

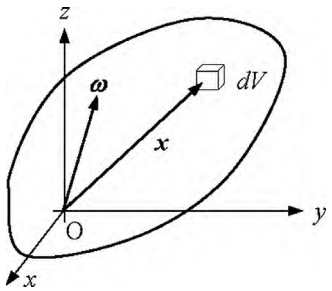


FIGURE 8.4

Typical volume element in a rotating rigid body.

Note that $\dot{\boldsymbol{x}}$ is the velocity of dm as viewed by a nonrotating observer translating with O; the velocity is $\dot{\boldsymbol{x}} = \boldsymbol{\omega} \times \boldsymbol{x}$. Therefore, the angular momentum of the rigid body is, with ρ as the mass density of the body,

$$\boldsymbol{H}_O = \int_V \rho \boldsymbol{x} \times (\boldsymbol{\omega} \times \boldsymbol{x}) dV \tag{8.31}$$

where

$\boldsymbol{\omega} = \omega_x \mathbf{i} + \omega_y \mathbf{j} + \omega_z \mathbf{k}$.
$\boldsymbol{x} = x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k}$.
$\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{k}$ are the unit vectors along the x-, y-, and z-axes, respectively.

Note that

$$\boldsymbol{\omega} \times \boldsymbol{x} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ \omega_x & \omega_y & \omega_z \\ x & y & z \end{vmatrix} = (z\omega_y - y\omega_z)\boldsymbol{i} + (x\omega_z - z\omega_x)\boldsymbol{j} + (y\omega_x - x\omega_y)\boldsymbol{k} \tag{8.32}$$

and

$$\begin{aligned} \boldsymbol{x} \times (\boldsymbol{\omega} \times \boldsymbol{x}) = &\left[(y^2 + z^2)\omega_x - xy\omega_y - xz\omega_z\right]\boldsymbol{i} \\ &+ \left[-yx\omega_x + (x^2 + z^2)\omega_y - yz\omega_z\right]\boldsymbol{j} \\ &+ \left[-zx\omega_x - zy\omega_y + (x^2 + y^2)\omega_z\right]\boldsymbol{k} \end{aligned} \tag{8.33}$$

Thus, Eq. 8.31 becomes

$$\begin{aligned} \boldsymbol{H}_O &= \boldsymbol{H}_x\boldsymbol{i} + \boldsymbol{H}_y\boldsymbol{j} + \boldsymbol{H}_z\boldsymbol{k} \\ &= (I_{xx}\omega_x + I_{xy}\omega_y + I_{xz}\omega_z)\boldsymbol{i} + (I_{yx}\omega_x + I_{yy}\omega_y + I_{yz}\omega_z)\boldsymbol{j} + (I_{zx}\omega_x + I_{zy}\omega_y + I_{zz}\omega_z)\boldsymbol{k} \end{aligned} \tag{8.34}$$

where I represents the mass moments of inertia; that is,

$$\begin{aligned} &I_{xx} = \int_V \rho(y^2 + z^2)dV, \quad I_{yy} = \int_V \rho(x^2 + z^2)dV, \quad I_{zz} = \int_V \rho(x^2 + y^2)dV \\ &I_{xy} = I_{yx} = -\int_V \rho xy dV, \quad I_{xz} = I_{zx} = -\int_V \rho xz dV, \quad I_{yz} = I_{zy} = -\int_V \rho yz dV \end{aligned} \tag{8.35}$$

If the x-, y-, and z-axes align with the principal axes of the rigid body, $I_{xy} = I_{yz} = I_{xz} = 0$. Therefore, Eq. 8.34 gives

$$H_x = I_{xx}\boldsymbol{\omega}_x,\ H_y = I_{yy}\boldsymbol{\omega}_y,\ \text{and}\ H_z = I_{zz}\boldsymbol{\omega}_z \tag{8.36}$$

In this case, Eq. 8.29 gives

$$\begin{aligned} M_x &= \dot{H}_x = I_{xx}\dot{\omega}_x \\ M_y &= \dot{H}_y = I_{yy}\dot{\omega}_y \\ M_z &= \dot{H}_z = I_{zz}\dot{\omega}_z \end{aligned} \tag{8.37}$$

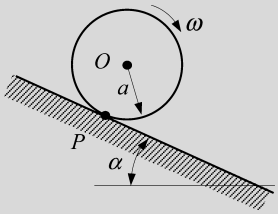
A simple example is presented next to illustrate the use of angular momentum for solving rigid-body dynamics problems.

EXAMPLE 8.4

A uniform circular cylinder of mass m, length L, and radius a rolls without slipping on a plane inclined at an angle α with the horizontal. Solve for the angular acceleration of the cylinder.

Solution

Taking the center of mass O as the reference point, we can use the rotational equations, Eqs 8.29 and 8.31, to derive the equation of motion for this rigid body. From the free-body diagram the external moment applied to the body is $M = Fa$, where F is an unknown friction force ensuring pure rolling.

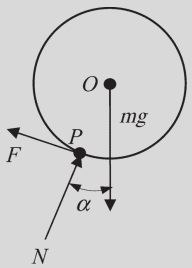


The cylinder rotates about its mass center O; hence, the angular momentum H_O, from Eq. 8.31, is

$$\begin{aligned} H_O &= \int_V \rho r(\omega r)\,dV = \int_0^a \rho r(\omega r)(2\pi r L)\,dr = \frac{1}{2}\rho\omega\pi a^4 L \\ &= \frac{1}{2}\rho a^2\omega\left(\pi a^2 L\right) = \frac{1}{2}ma^2\omega = \frac{1}{2}I\dot{\theta} \end{aligned} \tag{8.38}$$

where the position vector $\boldsymbol{x}$ becomes radius r. Hence, from Eq. 8.29 we have $Fa = \frac{1}{2}I\ddot{\theta} = \frac{1}{2}ma^2\ddot{\theta}$; therefore,

$$F = \frac{1}{2}ma\ddot{\theta} \tag{8.39}$$



EXAMPLE 8.4—cont'd

Since F is unknown, we need one more equation, which can be obtained from Newton's second law. Applying the law along the inclined plane, we have

$$mg \sin \alpha - F = ma\ddot{\theta} \tag{8.40}$$

If we bring $F = \frac{1}{2}ma\ddot{\theta}$ into Eq. 8.40, we have $mg \sin \alpha = \frac{3}{2}ma\ddot{\theta}$; hence,

$$\ddot{\theta} = \frac{2g \sin \alpha}{3a} \tag{8.41}$$

Solving Eq. 8.41 for angular position and velocity is straightforward.

The energy method is also applicable to rigid-body problems. For a rigid body in motion, the kinetic energy is both translational and rotational:

$$T = \frac{1}{2}m\dot{\boldsymbol{x}}^2 + \frac{1}{2}\boldsymbol{\omega}\cdot\boldsymbol{H} \tag{8.42}$$

in which both the velocity $\dot{\boldsymbol{x}}$ and the angular momentum $\boldsymbol{H}$ are calculated at the mass center of the body. Note that you may apply the energy method to Example 8.4 to obtain the same equation of motion. Here is how to proceed.

The total kinetic energy of the rolling cylinder is, according to Eq. 8.42,

$$T = \frac{1}{2}m(a\dot{\theta})^2 + \frac{1}{2}\dot{\theta}\left(\frac{1}{2}ma^2\dot{\theta}\right) = \frac{3}{4}ma^2\dot{\theta}^2$$

On the other hand, the potential energy is $U = -mg(\theta a \sin \alpha)$. The Lagrangian is then

$$L = T - U = \frac{3}{4}ma^2\dot{\theta}^2 + mg\theta a \sin \alpha \tag{8.43}$$

Applying Eq. 8.8, we have

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} = \frac{d}{dt}\left(\frac{3}{2}ma^2\dot{\theta}\right) - mga \sin \alpha = \frac{3}{2}ma^2\ddot{\theta} - mga \sin \alpha = 0 \tag{8.44}$$

Thus, the result of Example 8.4 is obtained.

The momentum equation and its derivative for the rolling cylinder in Example 8.4 are straightforward since the cylinder only rotates in one direction and rotates along its principal axis, which passes through its mass center.

8.2.3 MULTIBODY KINEMATIC ANALYSIS

Multibody kinematic analysis involves formulating equations of motion and solving them for position, velocity, and acceleration of individual bodies in the system in time. Such an analysis is important for general mechanism analysis and design, particularly for workspace analysis and robotics, where position, velocity, and acceleration of the moving components must be known in order to assess the functionality and performance of the mechanical system.

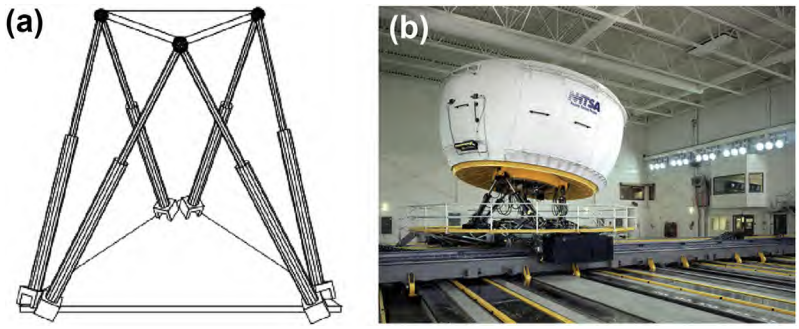


FIGURE 8.5

Stewart platform: (a) schematic view and (b) the University of Iowa's National Advanced Driving Simulator.

One of the most famous mechanisms commonly mentioned in spatial kinematic analysis is the Stewart platform, as shown in Figure 8.5(a). The Stewart platform mechanism, originally suggested by Stewart (1965), is a parallel kinematic structure that can be used as a basis for controlled motion with six degrees of freedom (DOF). The mechanism itself consists of a stationary platform (base platform) and a mobile platform that are connected via six legs (struts) mounted on universal joints. The legs have a built-in mechanism that allows changing the length of each individual leg. The desired position and orientation of the mobile platform are achieved by combining the lengths of the six legs, transforming the six transitional DOF into three positional (displacement vector) DOF and three orientation DOF (angles of rotation of a rigid body in space). The advantage of the Stewart platform is six degrees of freedom and a split-hair accuracy of mobile platform positioning. This makes Stewart platform widely usable in robot-building infrastructures. For example, such a parallel kinematic structure is used in the National Advanced Driving Simulator (NADS) shown in Figure 8.5(b).

In this section, instead of analyzing complex mechanisms such as the Stewart platform, which involves complex mathematical formulations, we discuss simple applications that can be solved analytically. One of the simplest and most useful mechanisms is the four-bar linkage, among which the slider-crank mechanism (Figure 8.6) can be seen in many applications such as internal combustion engines and oil-well-drilling equipment. For the internal combustion engine, the mechanism is driven by a firing load that pushes the piston, converting the reciprocal motion into rotational motion at the crank. In drilling equipment, a torque is applied at the crank. The rotational motion is converted to reciprocal motion at the drill bit that digs into the ground. Note that in all cases the length of the crank must be smaller than that of the rod to allow the mechanism to operate. This is commonly referred to as Grashof's law. Also in this section we briefly review the analytical methods

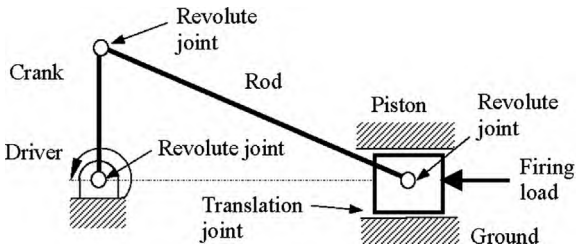


FIGURE 8.6

Schematic view of a slider-crank mechanism.

employed for multibody kinematic analysis using the slider-crank example. The methods reviewed should be applicable to more complex linkages.

The slider-crank mechanism consists of four bodies: crank, rod, slider, and the ground that is fixed to the reference frame. There are four joints—three revolute and one translation—defined among the bodies to restrain the relative motion between them. Assuming that the movable bodies only move on the plane, there are three degrees of freedom (DOF) for each one: two translations (along the x- and y-coordinates) and one rotation (along the z-axis). Also, because a planar revolute joint constrains translational motion of the body, allowing only rotational degrees of freedom, it removes two DOF. Similarly, a translation joint allows a body to move only along the translational direction, again removing two DOF. Hence, the total degrees of freedom of the planar slider-crank mechanism, or the so-called Gruebler's count, are

$$\begin{aligned} &3(\text{Moving bodies}) \times 3(\text{DOFs/body}) - 3(\text{revolute joints}) \\ &\quad \times 2(\text{DOFs/revolute}) - 1(\text{translation joint}) \times 2(\text{DOFs/translation}) = 9 - 8 = 1 \end{aligned} \tag{8.45}$$

where nb is the number of bodies in the system, and f_1 and f_2 are the DOF that revolute and translation joints remove, respectively. The free degree of freedom, which is commonly assumed as the rotation of the crank or the translation of the slider, can be driven by a rotation or translational motor. Once such a motor is added, the overall degrees of freedom of the system become zero. Note that for kinematic analysis, Gruebler's count of the mechanism must be equal to zero, excluding redundant DOF. More on the Gruebler's count will be discussed in Section 8.4.1.

There are three methods for solving kinematics of mechanism, which are commonly found in a textbook on mechanism design. They are the relative velocity or graphical method, the instant center method, and the analytical method based on complex variables. The following example illustrates the position and velocity computation for the slider-crank mechanism using the analytical method or, more specifically, the complex variable method.

EXAMPLE 8.5

Conduct a kinematic analysis for the slider-crank mechanism shown in Figure 8.6.

Solution

In kinematic analysis, forces and torque are not involved. Because all bodies are assumed massless, mass properties defined for them do not influence the analysis results.

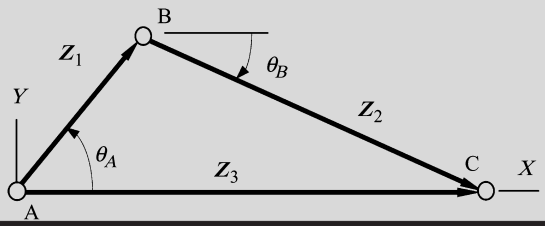
The slider-crank mechanism is a planar kinematic analysis problem. A vector plot that represents the positions of joints of the planar mechanism is shown in the following figure. The vector plot serves as the first step in computing the mechanism's position, velocity, and accelerations.

The position equations of the system can be described by the following vector summation:

$$\mathbf{Z}_1 + \mathbf{Z}_2 = \mathbf{Z}_3 \tag{8.46}$$

where

$\mathbf{Z}_1 = Z_1 \cos\theta_A + iZ_1 \sin\theta_A = Z_1 e^{i\theta A}$
$\mathbf{Z}_2 = Z_2 \cos\theta_B + iZ_2 \sin\theta_B = Z_2 e^{i\theta B}$
$\mathbf{Z}_3 = Z_3$, since θ_c is always 0



Continued

EXAMPLE 8.5—cont'd

The real and imaginary parts of Eq. 8.46, corresponding to the *X*- and *Y*-components of the vectors, can be written as

$$Z_1 \cos\theta_A + Z_2 \cos\theta_B = Z_3 \tag{8.47a}$$

$$Z_1 \sin\theta_A + Z_2 \sin\theta_B = 0 \tag{8.47b}$$

In Eqs 8.47a and 8.47b, Z_1, Z_2, and θ_A are given. We are solving for Z_3 and θ_B. These equations are nonlinear. Solving them directly for Z_2 and θ_B is not straightforward. Instead, we calculate Z_3 first, using trigonometric relations:

$$Z_2^2 = Z_1^2 + Z_3^2 - 2Z_1 Z_3 \cos\theta_A$$

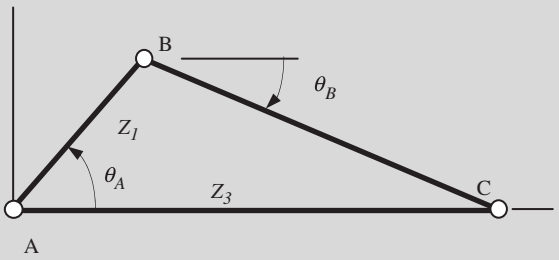
Thus,

$$Z_3^2 - 2Z_1 \cos\theta_A Z_3 + Z_1^2 - Z_2^2 = 0$$

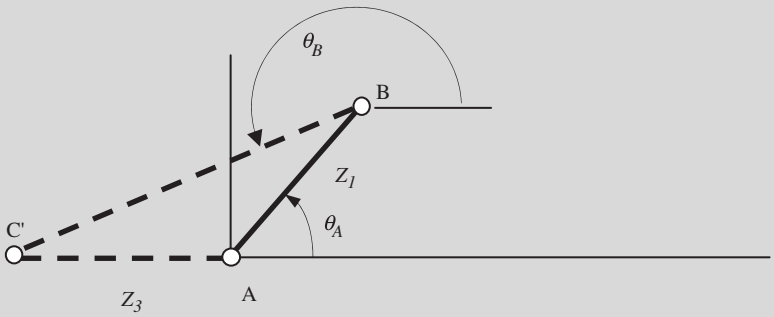
Solving Z_3 from the preceding quadratic equation, we have

$$Z_3 = \frac{2Z_1 \cos\theta_A \pm \sqrt{(2Z_1 \cos\theta_A)^2 - 4(Z_1^2 - Z_2^2)}}{2} \tag{8.48}$$

where two solutions of Z_3 represent the two possible configurations of the mechanism shown in the following figure. Note that point C can be either at C or C′ for any given Z_1 and θ_A.



Configurations 1:$\theta_B < 180°$



Configurations 2:$\theta_B > 180°$

EXAMPLE 8.5—cont'd

From Eq. 8.47b, θ_B can be solved by

$$\theta_B = \sin^{-1}\left(\frac{-Z_1 \sin\theta_A}{Z_2}\right) \tag{8.49}$$

Similarly, θ_B has two possible solutions corresponding to vector $\mathbf{Z}_3$.
Taking derivatives of Eqs 8.47a and 8.47b with respect to time, we have

$$-Z_1 \sin\theta_A\dot{\theta}_A - Z_2 \sin\theta_B\dot{\theta}_B = \dot{Z}_3 \tag{8.50a}$$

$$Z_1 \cos\theta_A\dot{\theta}_A + Z_2 \cos\theta_B\dot{\theta}_B = 0 \tag{8.50b}$$

where $\dot{\theta}_A = \frac{d\theta_A}{dt} = \omega_A$ is the angular velocity of the rotation driver, which is a constant. Note that Eqs 8.50a and 8.50b are linear functions of $\dot{Z}_3$ and $\dot{\theta}_B$. Rewrite the equations in matrix form:

$$\begin{bmatrix} Z_2 \sin\theta_B & 1 \\ Z_2 \cos\theta_B & 0 \end{bmatrix}\begin{bmatrix} \dot{\theta}_B \\ \dot{Z}_3 \end{bmatrix} = \begin{bmatrix} -Z_1 \sin\theta_A\dot{\theta}_A \\ -Z_1 \cos\theta_A\dot{\theta}_A \end{bmatrix} \tag{8.51}$$

Equation 8.51 can be solved by

$$\begin{aligned}\begin{bmatrix} \dot{\theta}_B \\ \dot{Z}_3 \end{bmatrix} &= \begin{bmatrix} Z_2 \sin\theta_B & 1 \\ Z_2 \cos\theta_B & 0 \end{bmatrix}^{-1}\begin{bmatrix} -Z_1 \sin\theta_A\dot{\theta}_A \\ -Z_1 \cos\theta_A\dot{\theta}_A \end{bmatrix} \\ &= \begin{bmatrix} -\dfrac{Z_1 \cos\theta_A\dot{\theta}_A}{Z_2 \cos\theta_B} \\ -\dfrac{Z_1\left(\cos\theta_B \sin\theta_A\dot{\theta}_A - \sin\theta_B \cos\theta_A\dot{\theta}_A\right)}{\cos\theta_B} \end{bmatrix}\end{aligned} \tag{8.52}$$

Therefore,

$$\dot{\theta}_B = -\frac{Z_1 \cos\theta_A\dot{\theta}_A}{Z_2 \cos\theta_B} \tag{8.53}$$

and

$$\dot{Z}_3 = Z_1\left(\tan\theta_B \cos\theta_A\dot{\theta}_A - \sin\theta_A\dot{\theta}_A\right) \tag{8.54}$$

8.2.4 MULTIBODY DYNAMIC ANALYSIS

A multibody system is used to model the dynamic behavior of interconnected rigid or flexible bodies, each of which may undergo large translational and rotational displacements. The systematic treatment of the dynamic behavior of interconnected bodies has led to a large number of important multibody formulations in the field of mechanics.

The simplest bodies or elements of a multibody system were treated by Newton's law, as briefly discussed earlier. Basically, the motion of bodies is described by their kinematic behavior. The dynamic behavior results from the equilibrium of applied forces and the rate of change in momentum. Nowadays, the term *multibody system* relates to a large number of engineering research fields, especially robotics and vehicle dynamics. As an important feature, multibody system dynamics

usually offers an algorithmic, computer-aided way to model, analyze, simulate, and optimize the arbitrary motion of possibly thousands of interconnected bodies.

The equations of motion are used to describe the dynamic behavior of a multibody system. Each multibody system formulation may lead to a different mathematical appearance of the equations of motion while the physics behind them remains the same. The motion of constrained bodies is described by equations that result basically from Newton's second law. These equations are written for the general motion of individual bodies with the addition of constraint conditions. Usually the equations of motion are derived from Newton and Euler equations or from Lagrange equations. We mentioned Newton's laws and Lagrange's equations; now we briefly discuss Euler equations, which govern the rotational motion of rigid bodies.

A rigid body rotates together with its body-fixed reference frame x-y-z at an angular velocity $\boldsymbol{\omega}$ with respect to the inertia (fixed) frame X-Y-Z, as shown in Figure 8.7. Note that we assume that the body-fixed reference frame also originates at the mass center of the rigid body. For a rotating body, the angular momentum $\boldsymbol{H}_o$ is most conveniently expressed in the body reference frame x-y-z, where point o is the origin. $\boldsymbol{H}_o$ can be written as

$$\boldsymbol{H}_o = H_x\boldsymbol{i} + H_y\boldsymbol{j} + H_z\boldsymbol{k} \tag{8.55}$$

where the unit vectors $\boldsymbol{i}$,$\boldsymbol{j}$, and $\boldsymbol{k}$ align and rotate with the x-, y-, and z-axes, respectively. From Eq. 8.29, we have

$$\boldsymbol{M}_o = \dot{\boldsymbol{H}}_o \tag{8.56}$$

where $\boldsymbol{M}_o$ is the vector of external moment applied to the body referring to the body reference frame x-y-z.

The derivative of $\boldsymbol{H}_o$ is

$$\begin{aligned}\dot{\boldsymbol{H}}_0 &= (\dot{H}_x\boldsymbol{i} + \dot{H}_y\boldsymbol{j} + \dot{H}_z\boldsymbol{k}) + (H_x\dot{\boldsymbol{i}} + H_y\dot{\boldsymbol{j}} + H_z\dot{\boldsymbol{k}}) \\ &= (\dot{\boldsymbol{H}})_r + (H_x\boldsymbol{\omega}\times\boldsymbol{i} + H_y\boldsymbol{\omega}\times\boldsymbol{j} + H_z\boldsymbol{\omega}\times\boldsymbol{k}) = (\dot{\boldsymbol{H}})_r + \boldsymbol{\omega}\times\boldsymbol{H}\end{aligned} \tag{8.57}$$

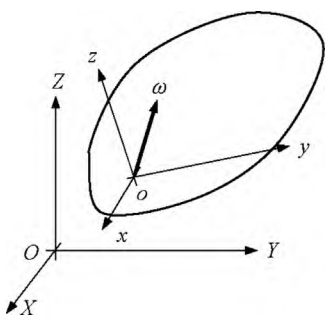


FIGURE 8.7

Rotating rigid body.

where $(\dot{\boldsymbol{H}})_r$ is the rate of change of the angular momentum with respect to o as viewed by an observer on the moving x-y-z coordinate system. We also assume that x, y, and z align with the principal axes of the body; hence, from Eq. 8.37,

$$(\dot{\boldsymbol{H}})_r = \dot{H}_x\boldsymbol{i} + \dot{H}_y\boldsymbol{j} + \dot{H}_z\boldsymbol{k} = I_{xx}\dot{\omega}_x\boldsymbol{i} + I_{yy}\dot{\omega}_y\boldsymbol{j} + I_{zz}\dot{\omega}_z\boldsymbol{k} \tag{8.58}$$

And the second term, after computing the cross-product and considering the principal axis assumption, is

$$\boldsymbol{\omega} \times \boldsymbol{H} = (I_{zz} - I_{yy})\omega_y\omega_z\boldsymbol{i} + (I_{xx} - I_{zz})\omega_x\omega_z\boldsymbol{j} + (I_{yy} - I_{xx})\omega_y\omega_x\boldsymbol{k} \tag{8.59}$$

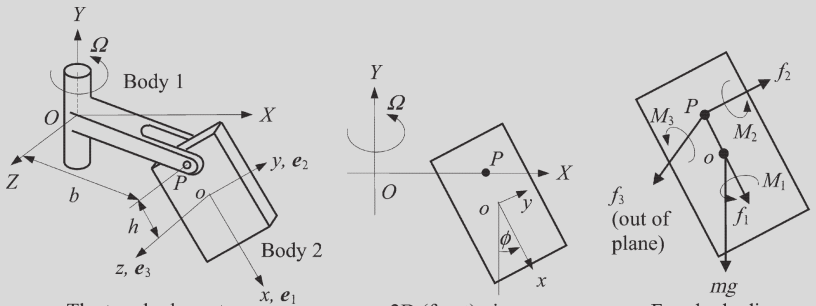
Combining Eqs 8.56 through 8.59, we have

$$\begin{aligned} M_{ox} &= I_{xx}\dot{\omega}_x + (I_{zz} - I_{yy})\omega_y\omega_z \\ M_{oy} &= I_{yy}\dot{\omega}_y + (I_{xx} - I_{zz})\omega_x\omega_z \\ M_{oz} &= I_{zz}\dot{\omega}_z + (I_{yy} - I_{xx})\omega_y\omega_x \end{aligned} \tag{8.60}$$

These are known as Euler's equations of motion. Example 8.6 illustrates a few details in solving a simple dynamic problem of a rigid-body system in rotation using the Euler equations.

EXAMPLE 8.6

A two-body system connected by a revolute (or pin) joint at point P is shown in the following figure. Body 1 rotates along the Y-axis at angular velocity Ω, and Body 2 rotates along the z-axis at angular velocity $\dot{\theta}$. Assume no friction at the pin joint between the two bodies. Note that point o is the mass center of Body 2 and the x-, y-, and z-axes align with the principal axes of Body 2. Formulate equations of motion that need to be solved for the rotation angle of Body 2 as well as the reaction forces (f_1, f_2, and f_3) and torque (M_1, M_2, and M_3) between the two bodies at the pin joint.



Solution

We formulate rotation equations of motion using Euler's equations first. Then we formulate equations for calculating reaction forces.

First, the angular velocity of Body 2 at its mass center o is

$$\boldsymbol{\omega}_o = \dot{\theta}\boldsymbol{e}_3 + \Omega\boldsymbol{j} \tag{8.61}$$

Note that in this example, unit vectors $\boldsymbol{i}, \boldsymbol{j}$, and $\boldsymbol{k}$ align with the X-, Y-, and Z-axes, respectively; similarly, unit vectors $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, and $\boldsymbol{e}_3$ align with the x-, y-, and z-axes, respectively.

Continued

EXAMPLE 8.6—cont'd

To apply Euler's equations, the angular velocity must be represented in the body-fixed reference frame (i.e., the x-y-z coordinate system). Hence, Eq. 8.61 becomes

$$\boldsymbol{\omega}_o = \dot{\phi}\boldsymbol{e}_3 + \Omega\left(-\cos\phi\boldsymbol{e}_1 + \sin\phi\boldsymbol{e}_2\right) = -\Omega\cos\phi\boldsymbol{e}_1 + \Omega\sin\phi\boldsymbol{e}_2 + \dot{\phi}\boldsymbol{e}_3 \tag{8.62}$$

and

$$\begin{aligned}\dot{\boldsymbol{\omega}}_0 &= \Omega\sin\phi\dot{\phi}\boldsymbol{e}_1 - \Omega\cos\phi\dot{\boldsymbol{e}}_1 + \Omega\cos\phi\dot{\phi}\boldsymbol{e}_2 + \Omega\sin\phi\dot{\boldsymbol{e}}_2 + \ddot{\phi}\boldsymbol{e}_3 + \dot{\phi}\dot{\boldsymbol{e}}_3\\ &= \left(\Omega\sin\phi\dot{\phi}\boldsymbol{e}_1 + \Omega\cos\phi\dot{\phi}\boldsymbol{e}_2 + \ddot{\phi}\boldsymbol{e}_3\right) - \Omega\cos\phi\boldsymbol{\omega}_0\times\boldsymbol{e}_1\\ &\quad + \Omega\sin\phi\boldsymbol{\omega}_0\times\boldsymbol{e}_2 + \phi\boldsymbol{\omega}_0\times\boldsymbol{e}_3\\ &= \Omega\sin\phi\dot{\phi}\boldsymbol{e}_1 + \Omega\cos\phi\dot{\phi}\boldsymbol{e}_2 + \ddot{\phi}\boldsymbol{e}_3\end{aligned} \tag{8.63}$$

As shown in the free-body diagram, the reaction forces f_1, f_2, and f_3, as well as moments M_1, M_2, and M_3 at the pin joint P, and the weight of Body 2 are the only external loads applied to the body. Hence, according to Euler's equations, the moment equations are

$$\sum M_x = I_{xx}\Omega\sin\phi\dot{\phi} + (I_{zz} - I_{yy})\Omega\sin\phi\dot{\phi} = M_1 \tag{8.64a}$$

$$\sum M_y = I_{yy}\Omega\cos\phi\dot{\phi} - (I_{xx} - I_{zz})\Omega\cos\phi\dot{\phi} = M_2 + hf_3 \tag{8.64b}$$

$$\sum M_z = I_{zz}\ddot{\phi} - (I_{yy} - I_{xx})\Omega^2\cos\phi\sin\phi\dot{\phi} = M_3 - hf_2 = -hf_2 \tag{8.64c}$$

where $M_3 = 0$, which is due to the fact that Body 2 is free to rotate along the z-axis at the pin joint.

Now for the translational motion, we apply Newton's second law to Body 2. We calculate the acceleration $\boldsymbol{a}_o$ of Body 2 at its mass center first. From particle dynamics,

$$\boldsymbol{a}_o = \boldsymbol{a}_p + \boldsymbol{a}_{o/p} = \boldsymbol{a}_p + \boldsymbol{\alpha}_o\times\boldsymbol{r} + \boldsymbol{\omega}_o\times\boldsymbol{\omega}_o\times\boldsymbol{r} \tag{8.65}$$

where $\boldsymbol{r}$ is the position vector from point P to o (i.e., $\boldsymbol{r} = h\boldsymbol{e}_1$), and

$$\boldsymbol{a}_p = -b\Omega^2\boldsymbol{i} = -b\Omega^2\left(\sin\phi\boldsymbol{e}_1 + \cos\phi\boldsymbol{e}_2\right) = -b\Omega^2\sin\phi\boldsymbol{e}_1 - b\Omega^2\cos\phi\boldsymbol{e}_2 \tag{8.66}$$

After vector multiplications and arranging terms, we have

$$\begin{aligned}\boldsymbol{a}_o &= \boldsymbol{a}_p + \dot{\boldsymbol{\omega}}_o\times\boldsymbol{r} + \boldsymbol{\omega}_o\times\boldsymbol{\omega}_o\times\boldsymbol{r}\\ &= \left(-b\Omega^2\sin\phi - h\Omega^2\sin^2\phi - h\dot{\phi}^2\right)\boldsymbol{e}_1\\ &\quad + \left(-b\Omega^2\cos\phi - h\Omega^2\cos\phi\sin\phi + h\ddot{\phi}\right)\boldsymbol{e}_2\\ &\quad + \left(-2h\Omega^2\dot{\phi}\cos\phi\right)\boldsymbol{e}_3\end{aligned} \tag{8.67}$$

Applying Newton's second law to Body 2, we have

$$\sum F_x = -m\left(-b\Omega^2\sin\phi - h\Omega^2\sin^2\phi - h\dot{\phi}^2\right) = f_1 + mg\cos\phi \tag{8.68a}$$

$$\sum F_y = m\left(-b\Omega^2\cos\phi - h\Omega^2\cos\phi\sin\phi - h\ddot{\phi}\right) = f_2 - mg\sin\phi \tag{8.68b}$$

$$\sum F_z = m\left(-2h\Omega^2\dot{\phi}\cos\phi\right) = f_3 \tag{8.68c}$$

These equations are too complex to solve by hand.

As shown in Example 8.6, multibody system dynamics problems are difficult to solve using analytical methods. In fact, the more critical issue in dynamics problems is the formulation of equations of motion. Certainly there are methods and principles such as Newton's law, conservation of energy, conservation of momentum, and so on. They must be applied to given problems in a case-by-case manner. Force and moment applied to bodies have to be identified, free-body diagrams have to be drawn, and then equations of motion can be formulated. Very often, we end up with equations of motion that are too complex to solve by hand, such as those in Example 8.6. There is no single method that is powerful and general enough to be consistently applied to formulate and solve general multibody dynamics problems.

A different approach that can be systematically applied to solve general problems must be employed for motion analysis of general mechanical systems. The computer-aided approach, which employs Newton's equations of motion governing the motion of rigid bodies, and constraint equations, governing the relative motion between bodies, is introduced in the next section. Formulation of the computer-aided method is uniform and consistent; therefore, it is more suitable for computer implementation. Equations of motion can be solved using numerical methods implemented on computers.

8.3 COMPUTER-AIDED METHODS

Large displacements and rotations that occur in the kinematic and dynamic performance of mechanical systems lead to nonlinear mathematical models that must be formulated and analyzed. The analytical complexity of nonlinear algebraic equations of kinematics and nonlinear differential equations of dynamics makes it impossible to obtain closed-form solutions in most applications.

The main objective of computer-aided methods is to create a systematic approach for both formulating and solving equations of motion that can be implemented on a digital computer. However, only if such a systematic approach is adopted can the burden of extensive analytical derivation be taken away from the shoulders of the engineer and delegated to the computer. The basic idea in computer-aided methods is introducing constraints that define relative motion between bodies in a multibody mechanical system, taking derivatives of the constraint equations to obtain velocity and acceleration equations, and then solving these equations using numerical methods. For dynamic analysis, Newton's law of motion is coupled with the constraint equations using Lagrange multipliers.

A thorough and in-depth discussion of the theory and mathematical formulation of computer-aided kinematic and dynamic analyses is beyond the scope of this book. Only a brief treatment is provided in this section. To simplify the mathematical expressions, the formulation of constraint equations and equations of motion focuses on mostly planar motion problems. This section aims at providing a short discussion on the theory and formulation of multibody systems with simple examples to help the reader understand the behind-the-scenes operations and to create models and use the computer tools more effectively. Several excellent references, such as Haug (1989), treat this subject with great details.

8.3.1 KINEMATIC ANALYSIS

We start with a basic formulation for planar kinematic analysis. Formulation of selected joints that are commonly found in motion analysis are included.

Any set of variables that uniquely specifies the position and orientation of all bodies in the mechanism—that is, the configuration of the mechanism—is a set of generalized coordinates $\boldsymbol{q} = [q_1, q_2, \ldots, q_{nc}]^{\mathrm{T}}$, where nc is the number of these coordinates. If a planar mechanism is made up of nb rigid bodies, the number of planar Cartesian generalized coordinates is $nc = 3nb$, since each planar body moves in two directions and rotates along the direction that is normal to the plane. Note that for spatial mechanisms, $nc = 6nb$ since each spatial body moves and rotates in three directions. Generalized coordinates may be independent (i.e., free to vary) or dependent (i.e., required to satisfy constraint equations).

A kinematic constraint between bodies imposes conditions on the relative motion between a pair of bodies. When these conditions are expressed as algebraic equations in terms of generalized coordinates, they are called *holonomic* kinematic constraint equations. A system of nh holonomic kinematic constraint equations can be expressed as

$$\boldsymbol{\Phi}^K(\boldsymbol{q},t) = \left[\boldsymbol{\Phi}_1^K(\boldsymbol{q},t), \boldsymbol{\Phi}_2^K(\boldsymbol{q},t), \ldots \boldsymbol{\Phi}_{nh}^K(\boldsymbol{q},t)\right]^{\mathrm{T}} = \mathbf{0} \tag{8.69}$$

Note that some or all the constraints may be stationary; that is, $\boldsymbol{\Phi}^K(\boldsymbol{q},t) = \boldsymbol{\Phi}^K(\boldsymbol{q}) = 0$.

If the constraints of Eq. 8.69 are independent, then the system is said to have $nc - nh$ degrees of freedom; in other words, DOF $= nc - nh$. To determine the motion of the system, we must either define additional driving constraints for the DOF so as to uniquely determine $\boldsymbol{q}(t)$ (kinematic analysis) or define forces that act on the system, in which case $\boldsymbol{q}(t)$ is the solution of equations of motion (dynamic analysis). If the independent driving constraints are specified for kinematic analysis, denoted

$$\boldsymbol{\Phi}^D(\boldsymbol{q},t) = 0 \tag{8.70}$$

then the configuration of the system as a function of time can be determined. Thus, the combined constraints of Eq. 8.69 and 8.70,

$$\boldsymbol{\Phi}(\boldsymbol{q},t) = \begin{bmatrix} \boldsymbol{\Phi}^K(\boldsymbol{q},t) \\ \boldsymbol{\Phi}^D(\boldsymbol{q},t) \end{bmatrix} = \mathbf{0} \tag{8.71}$$

can be solved for $\boldsymbol{q}(t)$. Such a system is called kinematically driven. Note that to guarantee solutions for Eq. 8.71, the size of $\boldsymbol{\Phi}$ (i.e., the number of constraint equations in it) must equal the number of generalized coordinates. In addition, all constraint equations in Eq. 8.71 must be independent. The very best way to test if the equation is solvable is that the Jacobian of $\boldsymbol{\Phi}_q$ must be nonsingular. Note that $\boldsymbol{\Phi}_q = \partial\boldsymbol{\Phi}/\partial\boldsymbol{q} = \boldsymbol{\Phi}_{,q}$. We use this shorthand notation throughout the chapter.

EXAMPLE 8.7

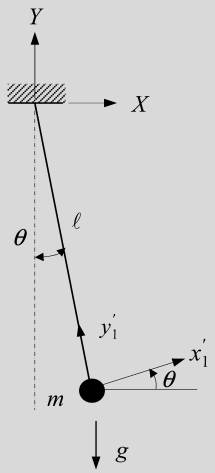
The same pendulum discussed in Example 8.1 is presented again to illustrate the formulation of particle kinematics using a computer-aided kinematic analysis method.

Solution

For the simple pendulum shown in the next figure, the kinematic constraint is described by the pair of equations,

$$\boldsymbol{\Phi}^K(\boldsymbol{q}) \equiv \begin{bmatrix} x_1 - \ell\sin\theta \\ y_1 + \ell\cos\theta \end{bmatrix} = \mathbf{0}$$

EXAMPLE 8.7—cont'd



where $\boldsymbol{q} = [x_1, y_1, \theta]^{\mathrm{T}}$. Note that x_1 and y_1 represent the particle location (or the origin of the $x_1' - y_1'$ frame). Clearly, the above equations can be solved for x_1 and y_1 as a function of θ, so one independent variable can specify the pendulum's motion. Thus, the system has one degree of freedom (DOF = 1).

To specify the motion of the pendulum, a driving constraint must be introduced, for example, specifying the time history of θ:

$$\boldsymbol{\Phi}^D(\theta, t) \equiv \theta - f(t) = 0 \tag{8.73}$$

Combining the previous two equations yields the system kinematic constraint equation:

$$\boldsymbol{\Phi}(\boldsymbol{q}, t) = \begin{bmatrix} \boldsymbol{\Phi}^K(\boldsymbol{q}) \\ \boldsymbol{\Phi}^D(\boldsymbol{q}, t) \end{bmatrix} = \begin{bmatrix} x_1 - \ell \sin\theta \\ y_1 + \ell \cos\theta \\ \theta - f(t) \end{bmatrix} = \mathbf{0} \tag{8.74}$$

This is a formulation for kinematic analysis. In reality, the pendulum rotates due to gravity, which falls into the scope of dynamic analysis. If we are interested in finding the reaction force or torque at the revolute joint, Eq. 8.74 is not sufficient. We need additional equations to calculate the reaction loads, in which dynamic analysis will be carried out.

If $\boldsymbol{\Phi}_{\boldsymbol{q}}$ is nonsingular—that is, $|\boldsymbol{\Phi}_{\boldsymbol{q}}| \neq 0$ at some value of $\boldsymbol{q}$ that satisfies Eq. 8.74—then Eq. 8.74 can be solved for $\boldsymbol{q}$ as a function of time. To test this condition, note that

$$|\boldsymbol{\Phi}_{\boldsymbol{q}}| = \begin{vmatrix} 1 & 0 & -\cos\theta \\ 0 & 1 & -\sin\theta \\ 0 & 0 & 1 \end{vmatrix} = 1 \tag{8.75}$$

Thus, from Eq. 8.74, once the rotation angle θ is given, the location of the mass center of the particle can be determined by solving for x_1 and y_1.

In general $\boldsymbol{q}$ is not known as an explicit function of time; it cannot be differentiated to obtain $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$. An alternative approach that is well suited to numerical computation is the chain rule of differentiation for evaluating derivatives of both sides of Eq. 8.71 with respect to time to obtain the velocity equation

$$\begin{aligned} \dot{\boldsymbol{\Phi}} &= \boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}} + \boldsymbol{\Phi}_t = \boldsymbol{0} \quad \text{or} \\ \boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}} &= -\boldsymbol{\Phi}_t \equiv \boldsymbol{\nu} \end{aligned} \tag{8.76}$$

If $\boldsymbol{\Phi}_{\boldsymbol{q}}$ is nonsingular, Eq. 8.76 can be solved for $\dot{\boldsymbol{q}}$ at discrete instants of time. Similarly, both sides of Eq. 8.76 can be differentiated again with respect to time, using the chain rule of differentiation to obtain

$$\boldsymbol{\Phi}_{\boldsymbol{q}}\ddot{\boldsymbol{q}} = -\left(\boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}}\right)_{\boldsymbol{q}}\dot{\boldsymbol{q}} - 2\boldsymbol{\Phi}_{\boldsymbol{q}t}\dot{\boldsymbol{q}} - \boldsymbol{\Phi}_{tt} \equiv \boldsymbol{\gamma} \tag{8.77}$$

Since $\boldsymbol{\Phi}_{\boldsymbol{q}}$ is nonsingular, Eq. 8.77 can be solved for $\ddot{\boldsymbol{q}}$ at discrete instants of time.

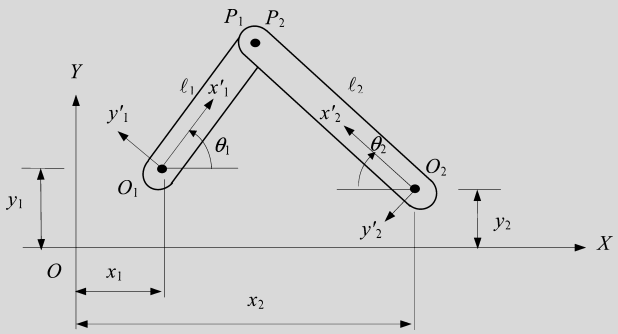
EXAMPLE 8.8

A slider-crank mechanism similar to that in Example 8.2 is presented to illustrate the computer-aided formulation for kinematic analysis. To simplify the mathematical presentation, a two-body system, composed of crank and rod, is assumed. The crank is driven by a driver of constant angular velocity ω.

Solution

There are two planar bodies in this system and one driving constraint; hence, we need $3nb - 1 = 5$ geometric constraint equations to uniquely define the kinematics of the system. First, Point O_1 on the crank (Body 1) coincides with the origin O of the X-Y frame; that is, $x_1 = y_1 = 0$. These are the first two constraint equations. The third equation is $y_2 = 0$ since Point O_2 on the rod (Body 2) must lie on the X-axis. The final two constraint equations are obtained by making Points P_1 and P_2 on the crank and rod, respectively, coincident:

$$\boldsymbol{r}^{P_1} \equiv \begin{bmatrix} x_1 + \ell_1 \cos\theta_1 \\ y_1 + \ell_1 \sin\theta_1 \end{bmatrix} = \begin{bmatrix} x_2 - \ell_2 \sin\theta_2 \\ y_2 + \ell_2 \cos\theta_2 \end{bmatrix} \equiv \boldsymbol{r}^{P_2} \tag{8.78}$$



Therefore, in terms of $\boldsymbol{q} = [x_1, y_1, \theta_1, x_2, y_2, \theta_2]^{\mathrm{T}}$, the system of kinematic constraint equations, combining with the driver, is

$$\boldsymbol{\Phi}(\boldsymbol{q}, t) \equiv \begin{bmatrix} x_1 \\ y_1 \\ y_2 \\ x_1 - x_2 + \ell_1 \cos\theta_1 + \ell_2 \sin\theta_2 \\ y_1 - y_2 + \ell_1 \sin\theta_1 - \ell_2 \cos\theta_2 \\ \theta_1 - \omega t \end{bmatrix} = 0 \tag{8.79}$$

EXAMPLE 8.8—cont'd

Solving for $\boldsymbol{q}$ yields

$$\boldsymbol{q} \equiv \begin{bmatrix} x_1 \\ y_1 \\ \theta_1 \\ x_2 \\ y_2 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \omega t \\ \ell_1 \cos\theta_1 + \ell_2 \sin\theta_2 \\ 0 \\ \pm\cos^{-1}\left(\dfrac{\ell_1 \sin\theta_1}{\ell_2}\right) \end{bmatrix} \tag{8.80}$$

in which ℓ_1 and ℓ_2 are known and $\dot{\theta}$ is given; hence, θ_1 can be solved, then θ_2, and finally x_2.

Taking derivatives on both sides of Eq. 8.79 with respect to time t yields

$$\boldsymbol{\Phi}_{\boldsymbol{q}}(\boldsymbol{q},t)\dot{\boldsymbol{q}} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & -\ell_1 \sin\theta_1 & -1 & 0 & \ell_2 \cos\theta_2 \\ 0 & 1 & \ell_1 \cos\theta_1 & 0 & -1 & \ell_2 \sin\theta_2 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{\theta}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \omega \end{bmatrix} = -\boldsymbol{\Phi}_t \tag{8.81}$$

Since $|\boldsymbol{\Phi}_{\boldsymbol{q}}| \neq 0$ (Exercise 8.6), $\dot{\boldsymbol{q}}$ can be solved using Eq. 8.81 as

$$\dot{\boldsymbol{q}} \equiv \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{\theta}_1 \\ \dot{x}_2 \\ \dot{y}_2 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \omega \\ -\ell_1\omega(\sin\theta_1 + \cos\theta_1 \cot\theta_2) \\ 0 \\ \dfrac{-\ell_1 \cos\theta_1 \omega}{\ell_2 \sin\theta_2} \end{bmatrix} \tag{8.82}$$

in which $\dot{\theta}_1$, θ_1, and θ_2 can be solved in Eq. 8.80, then $\dot{x}_2$, and finally $\dot{\theta}_2$.

Taking a derivative on both sides of Eq. 8.81 with respect to time t yields

$$\boldsymbol{\Phi}_{\boldsymbol{q}}(\boldsymbol{q},t)\ddot{\boldsymbol{q}} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & -\ell_1 \sin\theta_1 & -1 & 0 & \ell_2 \cos\theta_2 \\ 0 & 1 & \ell_1 \cos\theta_1 & 0 & -1 & \ell_2 \sin\theta_2 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{y}_1 \\ \ddot{\theta}_1 \\ \ddot{x}_2 \\ \ddot{y}_2 \\ \ddot{\theta}_2 \end{bmatrix} \tag{8.83}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dot{\theta}_1^2 \ell_1 \cos\theta_1 + \dot{\theta}_2^2 \ell_2 \sin\theta_2 \\ \dot{\theta}_1^2 \ell_1 \sin\theta_1 - \dot{\theta}_2^2 \ell_2 \cos\theta_2 \\ 0 \end{bmatrix}$$

$\ddot{\boldsymbol{q}}$ can be solved using Eq. 8.83. This is left as an exercise (Exercise 8.7).

With the Cartesian generalized coordinates (or absolute coordinates)—that is, position and orientation of every single body in the system—a large system of equations is obtained. For example, for a slider-crank mechanism that consists of three moving bodies, shown in Figure 8.8(a), the absolute coordinates are

$$\boldsymbol{q} = \left[x_1, y_1, \theta_1, x_2, y_2, \theta_2, x_3, y_3, \theta_3\right]^{\mathrm{T}} \tag{8.84}$$

However, using Cartesian generalized coordinates, the constraint equations can easily be formulated automatically, which is ideal for implementation on computers.

There is a simpler set of generalized coordinates that can be chosen for problem formulation. For the system shown in Figure 8.8(b), the joint coordinates that correspond to kinematic joints in system can be chosen as

$$\boldsymbol{q} = \left[\theta_A, \theta_B, Z_3\right]^{\mathrm{T}} \tag{8.85}$$

where

θ_A and θ_B correspond to the rotation DOF of the two revolute joints A and B, respectively.
Z_3 represents the translation DOF of the slider joint at point C.

Note that the parameters defined in Eq. 8.85 are consistent with those of Example 8.5.

Assuming the rotation angle of the crank is prescribed by a driving constraint $\theta_A = f(t) = \omega t$, the constraint equations of the system can be formulated in terms of $\boldsymbol{q} = \left[\theta_A, \theta_B, Z_3\right]^{\mathrm{T}}$ as

$$\boldsymbol{\Phi}(\boldsymbol{q}, t) = \begin{bmatrix} \ell_1 \cos\theta_A + \ell_2 \cos\theta_B - Z_3 \\ \ell_1 \sin\theta_A + \ell_2 \sin\theta_B \\ \theta_A - \omega t \end{bmatrix} = 0 \tag{8.86a}$$

The velocity and acceleration equations can be obtained, respectively, as

$$\boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}} = \begin{bmatrix} -\ell_1 \sin\theta_A & -\ell_2 \sin\theta_B & -1 \\ \ell_1 \cos\theta_A & \ell_2 \cos\theta_B & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_A \\ \dot{\theta}_B \\ \dot{Z}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} = -\boldsymbol{\Phi}_t \tag{8.86b}$$

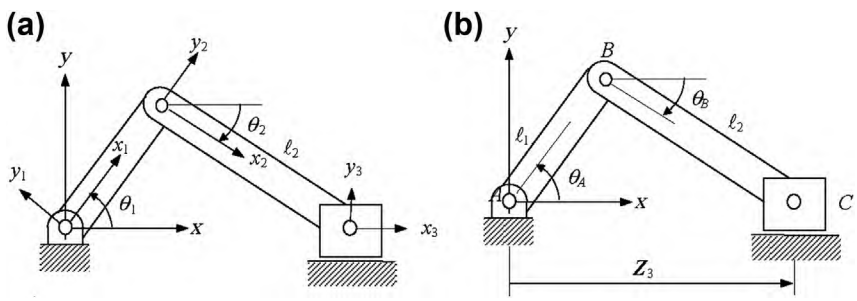


FIGURE 8.8

A slider-crank mechanism, (a) with Cartesian generalized coordinate system, and (b) with joint (relative) coordinates.

and

$$\boldsymbol{\Phi}_{\boldsymbol{q}}(\boldsymbol{q},t)\dot{\boldsymbol{q}} = \begin{bmatrix} -\ell_1 \sin\theta_A & -\ell_2 \sin\theta_B & -1 \\ \ell_1 \cos\theta_A & \ell_2 \cos\theta_B & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_A \\ \ddot{\theta}_B \\ \ddot{Z}_3 \end{bmatrix} = \begin{bmatrix} \ell_1 \cos\theta_A \dot{\theta}_A^2 + \ell_2 \cos\theta_B \dot{\theta}_B^2 \\ \ell_1 \sin\theta_A \dot{\theta}_A^2 + \ell_2 \sin\theta_B \dot{\theta}_B^2 \\ 0 \end{bmatrix} \equiv \boldsymbol{\gamma} \quad (8.86c)$$

Solving Eqs 8.86a, 8.86b, and 8.86c, the solutions of $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$ are, respectively,

$$\begin{bmatrix} \theta_A \\ \theta_B \\ Z_3 \end{bmatrix} = \begin{bmatrix} \omega t \\ \sin^{-1}\left(-\ell_1 \sin\theta_A/\ell_2\right) \\ \ell_1 \cos\theta_A + \ell_2 \cos\theta_B \end{bmatrix} = \begin{bmatrix} \omega t \\ \sin^{-1}\left(-\ell_1 \sin\theta_A/\ell_2\right) \\ \ell_1 \cos\theta_A \pm \sqrt{\ell_2^2 - (\ell_1 \sin\theta_A)^2} \end{bmatrix} \quad (8.87a)$$

$$\begin{bmatrix} \dot{\theta}_A \\ \dot{\theta}_B \\ \dot{Z}_3 \end{bmatrix} = \begin{bmatrix} \omega \\ -\dfrac{\ell_1 \cos\theta_A}{\ell_2 \cos\theta_B}\dot{\theta}_A \\ -\ell_1 \sin\theta_A \dot{\theta}_A + \ell_2 \sin\theta_B \dot{\theta}_B \end{bmatrix} = \begin{bmatrix} \omega \\ -\dfrac{\ell_1 \cos\theta_A}{\ell_2 \cos\theta_B}\dot{\theta}_A \\ \ell_1 \dot{\theta}_A \left(\tan\theta_B \cos\theta_A - \sin\theta_A\right) \end{bmatrix} \quad (8.87b)$$

$$\begin{bmatrix} \ddot{\theta}_A \\ \ddot{\theta}_B \\ \ddot{Z}_3 \end{bmatrix} = \begin{bmatrix} 0 \\ \left(\ell_1 \sin\theta_A \dot{\theta}_A^2 + \ell_2 \sin\theta_B \dot{\theta}_B^2\right) \Big/ (\ell_2 \cos\theta_B) \\ -\ell_1 \cos\theta_A \dot{\theta}_A^2 - \ell_2 \cos\theta_B \dot{\theta}_B^2 - \ell_2 \sin\theta_B \ddot{\theta}_B \end{bmatrix} \quad (8.87c)$$

Note that the results in Eqs 8.87a and 8.87b are identical to those in Example 8.5. Exercise 8.7 (at the end of the chapter) will indicate if the results match with Eq. 8.87c for accelerations.

8.3.2 KINEMATIC JOINTS

Kinematic joints (or simply *joints*) are critical parts of a mechanism. The resultant motion in operating a mechanism is largely determined by the kinematic joints connecting the members of the mechanism. The kinematic joints allow motion in some directions and constrain it in others. The types of motion allowed and constrained are related to the characteristics and intended use of the joint, which can be usually characterized by the degrees of freedom it allows.

For a planar mechanism, there are two kinds of joints: planar J1 joints that allow one DOF (restrict two DOF); and planar J2 joints that allow two DOF (restrict one DOF). The revolute and slider joints discussed before are J1 joints and the pin-in-slot is a J2 joint, as shown in Figure 8.9. Three-dimensional or spatial joints are classified into two categories based on the type of contact between the two members making a joint: lower pair joint and higher pair joint. The contact can be point, line, or area. A third category of kinematic joint comprises the joints formed by combining two or more lower pair and/or higher pair joints. Such joints are termed compound joints.

The two members forming a lower pair joint have area contact between the two mating surfaces. The contact stress is thus smaller for lower pair joints as compared to higher pair joints. Lower pair joints have a long service life because the wear and stress are spread over a larger surface area of

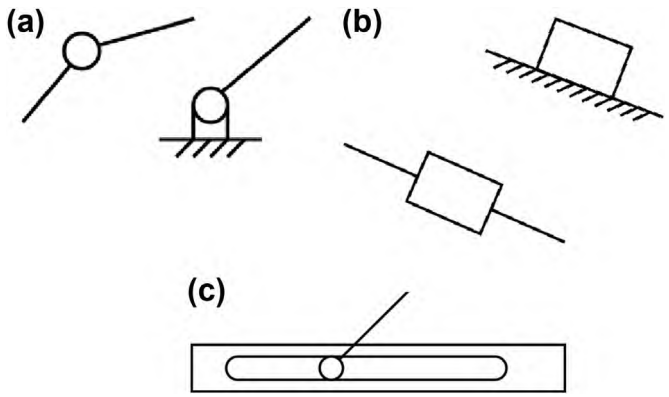


FIGURE 8.9

Planar joints. (a) J1: revolute/hinge/pin joint, (b) J1: prismatic/slider, and (c) J2: pin-in-slot.

contact, and they allow better lubrication. The degrees of freedom for a lower pair joint are usually fewer, as the requirement for area contact between the members constrains the joint geometry.

Some of the lower pair joints we commonly see in mechanical systems are shown in Figure 4.9. Different joints allow different kinds of motion. For example, a revolute (or hinge or pin) joint allows the rotation of one rigid body with respect to another rigid body about a common axis. Therefore, a revolute joint eliminates a total of five DOF: three translational and two rotational.

The contact between the two members of a higher pair joint has point or line geometry. The contact stress for a higher pair joint is large because the contact area is very small. If there is pure rolling contact between the members, then there is no relative sliding between the contact surfaces and thus friction and wear are minimal. The number of degrees of freedom for a higher pair joint can be large as the point or line contact allows for less constrained motion of members. A cam-follower, as shown in Figure 8.10, is a good example of higher pair joints, where a line contact is observed between the cam and the follower. A cam-follower allows two DOF, one rotational and one translational, along the center axes of the cam and the followers that are in parallel.

Lower pair and/or higher pair joints are combined as per the design requirement to obtain compound joints. Compound joints composed of higher pair joints can be kinematically equivalent to lower pair joints and vice versa. By such combinations desirable features from the combining joints are retained to achieve robust joints.

A good example of a composite joint is a ball or roller bearing. The actual members in contact are balls or rollers with the inner and outer face. This is a rolling contact, which is a higher pair. However, the overall joint has the motion geometry of a revolute joint, a lower pair. A ball bearing has the low friction properties of rolling contacts and the high load capacity of revolute joints. Ball or roller bearings are kinematically equivalent to simple revolute joints. Another example is the universal joint shown in Figure 4.9(f). This is a combination of revolute joints and has two rotational degrees of freedom.

Planar joints are much simpler to define mathematically than their counterparts in space. For example, a planar revolute joint allows relative rotation at a point P that is common to Bodies i and j, as shown in Figure 8.11(a). If one body is held fixed, the other body has only a single rotational DOF. Thus, a planar resolute joint eliminates two DOF from the pair. This joint is defined by locating point

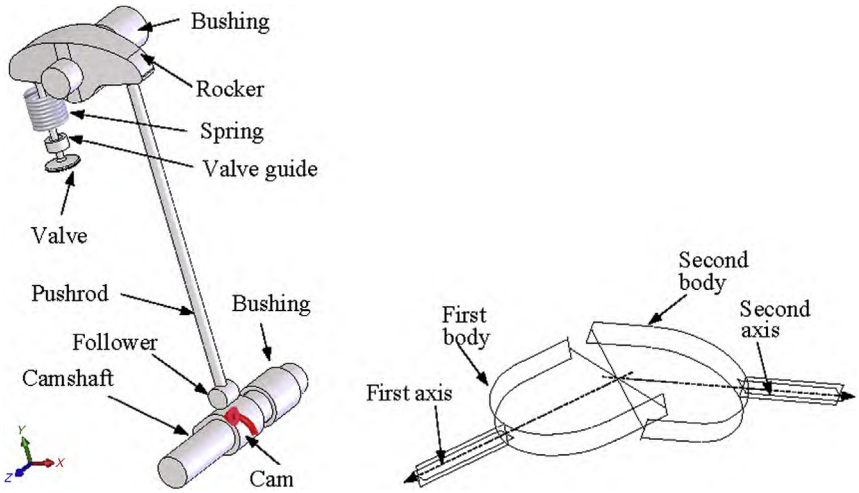


FIGURE 8.10

Higher pair joint: a cam-follower in the mechanism of an engine inlet or outlet valve.

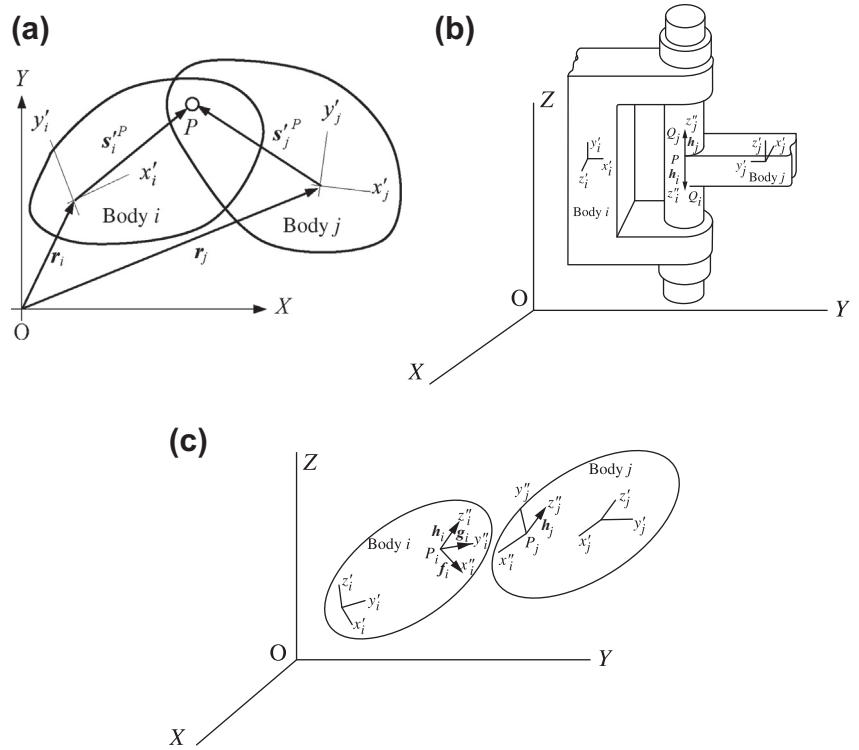


FIGURE 8.11

Mathematical formulation of a revolute joint: (a) planar revolute joint, (b) spatial revolute joint, and (c) dot-1 constraint.

P_i on Body i by $\boldsymbol{s}_i'^P$ in the $x_i' - y_i'$ frame (fixed to body i) and locating P_j on Body j by $\boldsymbol{s}_j'^P$ in the $x_j' - y_j'$ frame (fixed to body j), respectively. The constraint equation can be defined as

$$\begin{aligned}\boldsymbol{\Phi}^{r(i,j)} &= \boldsymbol{r}_i + \boldsymbol{s}_i^p - \boldsymbol{r}_j - \boldsymbol{s}_j^p \\ &= \boldsymbol{r}_i + \boldsymbol{A}_i \boldsymbol{s}_i'^P - \boldsymbol{r}_j - \boldsymbol{A}_j \boldsymbol{s}'^P_j\end{aligned} \tag{8.88}$$

where $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$ are the transformation matrices that transform position vectors $\boldsymbol{s}_i'^P$ and $\boldsymbol{s}_j'^P$ from their respective frames $x_i' - y_i'$ and $x_j' - y_j'$ to the global frame X-Y, respectively. Note that there are two equations in Eq. 8.88; therefore, motion is constrained in both the X- and Y-directions. Also note that the constraint equations of Eq. 8.78 in Example 8.8 define a revolute joint at points P_1 and P_2 of the crank and rod, respectively.

A spatial revolute joint between Bodies i and j allows relative rotation about a common axis, but precludes relative translation along this axis, as shown in Figure 8.11(b). To define the revolute joint, the joint center is located on Bodies i and j by points P_i and P_j. The axis of relative rotation is defined in Bodies i and j by points Q_i and Q_j and hence unit vectors $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ along the respective z′′-axes of the joint reference frames. The remaining joint definition frame axes are defined at the convenience of the designer. The analytical formulation of the revolute joint is that points P_i and P_j coincide and that body-fixed vectors $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ are parallel, leading to the constraint equations

$$\boldsymbol{\Phi}^s\left(P_i, P_j\right) = \boldsymbol{r}_i + \boldsymbol{A}_i \boldsymbol{s}_i'^P - \boldsymbol{r}_j - \boldsymbol{A}_j \boldsymbol{s}_j'^P = \boldsymbol{0} \tag{8.89a}$$

$$\boldsymbol{\Phi}^p\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) = \begin{bmatrix} \boldsymbol{\Phi}^d\left(\boldsymbol{f}_i, \boldsymbol{h}_j\right) \\ \boldsymbol{\Phi}^d\left(\boldsymbol{g}_i, \boldsymbol{h}_j\right) \end{bmatrix} = \boldsymbol{0} \tag{8.89b}$$

There are three scalar equations in Eq. 8.89a, which eliminate three relative translational DOF between Bodies i and j at points P_i and P_j. Eq. 8.89b defines a parallel constraint, consisting of two so-called dot-1 constraints. A dot-1 constraint is defined by a dot product of two perpendicular vectors. In this case, $\boldsymbol{f}_i$ is perpendicular to $\boldsymbol{h}_j$ and $\boldsymbol{g}_i$ is perpendicular to $\boldsymbol{h}_j$. Also, as shown in Figure 8.11(c), $\boldsymbol{h}_i$ is perpendicular to both $\boldsymbol{f}_i$ and $\boldsymbol{g}_i$; therefore, $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ are in parallel. There are, overall, five constraint equations in Eqs 8.89a and 8.89b; all are independent and allow only rotation along $\boldsymbol{h}_i$ or $\boldsymbol{h}_j$—thus, a revolute joint in space. The DOF that each spatial joint eliminates are summarized in Table 4.5.

8.3.3 MULTIBODY DYNAMIC ANALYSIS

Consider mechanical systems that are made up of a collection of rigid bodies in a plane, with kinematic constraints between them. The variational approach commonly found in dynamics textbooks is employed to formulate differential equations of motion. The key idea is to couple the differential equations of motion with the kinematic constraint equations by introducing Lagrange multipliers to account for the constraints.

The variational equation of motion for a rigid body in a plane can be formulated as

$$\delta \boldsymbol{r}^{\mathrm{T}}\left[m\ddot{\boldsymbol{r}} - \boldsymbol{F}\right] + \delta\theta^{\mathrm{T}}\left[J'\ddot{\theta} - n\right] = 0 \tag{8.90}$$

where

$\boldsymbol{r}$ is the position vector to the mass center of the body.
$\boldsymbol{F}$ and n are external forces and torque, respectively.

J' is the polar moment of inertia at the centroid of the body referring to the body-fixed reference frame x'-y' (see Figure 8.11(a)).
$\delta \boldsymbol{r}$ and $\delta\theta$ are the virtual displacements and rotation of the rigid body, respectively.

The body-fixed reference frame x'-y' is defined at the centroid of the body.
Equation 8.90 can be written as

$$\delta \boldsymbol{q}^{\mathrm{T}} \left[\boldsymbol{M}\ddot{\boldsymbol{q}} - \boldsymbol{Q} \right] = 0 \tag{8.91}$$

where

$\boldsymbol{q} = [\boldsymbol{r}, \theta]^{\mathrm{T}}$ is the vector of generalized coordinates.
$\boldsymbol{Q} = [\boldsymbol{F}, n]^{\mathrm{T}}$ is the vector of generalized forces.
$\boldsymbol{M}$ is the diagonal mass matrix consisting of mass m and moment of inertia J' for the rigid body.

The variational equations of motion for each Body i in a planar multibody system of nb bodies, given by Eq. 8.91, may be summed to obtain the system variational equations of motion:

$$\sum_{i=1}^{nb} \delta \boldsymbol{q}_i^{\mathrm{T}} \left[\boldsymbol{M}_i \ddot{\boldsymbol{q}}_i - \boldsymbol{Q}_i \right] = 0 \tag{8.92}$$

Redefine $\boldsymbol{q}$, $\boldsymbol{M}$, and $\boldsymbol{Q}$ as the composite state variable vector, composite mass matrix, and composite vector of generalized forces, respectively:

$$\boldsymbol{q} = \left[\boldsymbol{q}_1^{\mathrm{T}}, \boldsymbol{q}_2^{\mathrm{T}}, \ldots, \boldsymbol{q}_{nb}^{\mathrm{T}} \right]^{\mathrm{T}} \tag{8.93a}$$

$$\mathbf{M} = \mathrm{diag}(\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_{nb}) \tag{8.93b}$$

$$\boldsymbol{Q} = \left[\boldsymbol{Q}_1^{\mathrm{T}}, \boldsymbol{Q}_2^{\mathrm{T}}, \ldots, \boldsymbol{Q}_{nb}^{\mathrm{T}} \right]^{\mathrm{T}} \tag{8.93c}$$

The variational equations of Eq. 8.93 can be written as

$$\delta \boldsymbol{q}^{\mathrm{T}} \left[\boldsymbol{M}\ddot{\boldsymbol{q}} - \boldsymbol{Q}^A \right] - \delta \boldsymbol{q}^{\mathrm{T}} \boldsymbol{Q}^C = 0 \tag{8.94}$$

where the generalized force vector $\boldsymbol{Q}$ has been separated into two parts, $\boldsymbol{Q}^A$ and $\boldsymbol{Q}^C$, representing the externally applied forces and constraint forces due to the joints defined between bodies.

Note that by Newton's law of action and reaction, if there is no friction in kinematic joints, constraint forces act perpendicular to contacting surfaces and are equal in magnitude. Thus, if attention is restricted to virtual displacements that are consistent with the constraints that act on the system, then the virtual work of all constraint forces is zero:

$$\delta \boldsymbol{q}^{\mathrm{T}} \boldsymbol{Q}^C = 0 \tag{8.95}$$

Hence, Eq. 8.94 becomes

$$\delta \boldsymbol{q}^{\mathrm{T}} \left[\boldsymbol{M}\ddot{\boldsymbol{q}} - \boldsymbol{Q}^A \right] = 0 \tag{8.96}$$

Now we introduce Lagrange multipliers $\boldsymbol{\lambda}$ to couple the equation of motion of Eq. 8.96 with the constraint equation of Eq. 8.71:

$$\left[\boldsymbol{M}\ddot{\boldsymbol{q}} - \boldsymbol{Q}^A \right]^{\mathrm{T}} \delta \boldsymbol{q} + \boldsymbol{\lambda}^T \boldsymbol{\Phi}_{\boldsymbol{q}} \delta \boldsymbol{q} = \left[\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{\lambda}^{\mathrm{T}} \boldsymbol{\Phi}_{\boldsymbol{q}} - \boldsymbol{Q}^A \right]^{\mathrm{T}} \delta \boldsymbol{q} = 0 \tag{8.97}$$

for arbitrary $\delta \boldsymbol{q}$. Therefore, the Lagrange multiplier form of the equations of motion is

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{\lambda}^{\mathrm{T}}\boldsymbol{\Phi}_{\boldsymbol{q}} - \boldsymbol{Q}^{A} = \boldsymbol{0} \tag{8.98}$$

In addition to these equations of motion, recall that the velocity and acceleration equations of Eqs 8.76 and 8.77, respectively, comprise the complete set of constrained equations of motion for the system.

Equations 8.98 and 8.77 may be written in matrix form as

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{\Phi}_{q}^{\mathrm{T}} \\ \boldsymbol{\Phi}_{q} & 0 \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Q}^{A} \\ \boldsymbol{\gamma} \end{bmatrix} \tag{8.99}$$

This is a mixed system of differential-algebraic equations (DAE) since no derivatives of the Lagrange multipliers $\boldsymbol{\lambda}$ appear. Note that the Lagrange multipliers are related to the reaction forces and torque at the joint. These reactions are critical for the structural integrity and durability of the product.

EXAMPLE 8.9

Derive the differential-algebraic equation for the pendulum from Example 8.7.

Solution

Successively differentiate the following constraint equations:

$$\boldsymbol{\Phi}(\boldsymbol{q}) \equiv \begin{bmatrix} x_1 - \ell \sin\theta \\ y_1 + \ell \cos\theta \end{bmatrix} = 0 \tag{8.100}$$

The velocity and acceleration equations are, respectively,

$$\boldsymbol{\Phi}_{\boldsymbol{q}}\dot{\boldsymbol{q}} = \begin{bmatrix} 1 & 0 & -\ell\cos\theta \\ 0 & 1 & -\ell\sin\theta \end{bmatrix} \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \\ \dot{\theta} \end{bmatrix} = \boldsymbol{0} \equiv \boldsymbol{\nu} \tag{8.101}$$

$$\boldsymbol{\Phi}_{\boldsymbol{q}}\ddot{\boldsymbol{q}} = \begin{bmatrix} -\ell\dot{\theta}^2 \sin\theta \\ \ell\dot{\theta}^2 \cos\theta \end{bmatrix} \equiv \boldsymbol{\gamma} \tag{8.102}$$

From Eqs 8.99 and 8.102, the differential-algebraic equations are

$$\begin{bmatrix} m & 0 & 0 & 1 & 0 \\ 0 & m & 0 & 0 & 1 \\ 0 & 0 & 0 & -\ell\cos\theta & -\ell\sin\theta \\ 1 & 0 & -\ell\cos\theta & 0 & 0 \\ 0 & 1 & -\ell\sin\theta & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x}_1 \\ \ddot{y}_1 \\ \ddot{\theta} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 0 \\ -mg \\ 0 \\ -\ell\dot{\theta}^2 \sin\theta \\ \ell\dot{\theta}^2 \cos\theta \end{bmatrix} \tag{8.103}$$

Solve Eq. 8.103 to obtain

EXAMPLE 8.9—cont'd

$$\begin{bmatrix} \ddot{x}_1 \\ \ddot{y}_1 \\ \ddot{\theta} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} -\sin\theta\left(\ell\dot{\theta}^2 + g\cos\theta\right) \\ -g\sin^2\theta + \ell\cos\theta\dot{\theta}^2 \\ -\frac{g}{\ell}\sin\theta \\ -m\ddot{x}_1 \\ -m\left(g + \ddot{y}_1\right) \end{bmatrix} \tag{8.104}$$

Note that the third equation in Eq. 8.104 is identical to that in Example 8.1, which was derived using Newton's method. Also, in this example, λ_1 and λ_2 are the reaction forces at the pin joint.

The approach just shown is much more general than those found in classical dynamics (some of the methods were discussed in Section 8.2). Any mechanical system that is characterized by bodies and kinematic joints can be formulated as a set of differential equations (Eq. 8.86) and as a mixed system of differential-algebraic equations, as shown in Eq. 8.99 for kinematic and dynamic analysis, respectively. Equations of motion can be solved using numerous numerical algorithms. Interested readers are referred to, for example, Haug (1989) and Kane and Levinson (1985) for a comprehensive discussion of this subject.

8.4 MOTION SIMULATION

The overall process of using computer tools for motion analysis consists of three main steps: model generation (or pre-processing), analysis (or simulation), and result visualization (or post-processing), as illustrated in Figure 8.12. Key entities that constitute a motion model include servomotors that drive the mechanism for kinematic analysis, external loads (force and torque), force entities such as spring and damper, and the mechanism's initial conditions. Most important, assembly mates must be properly

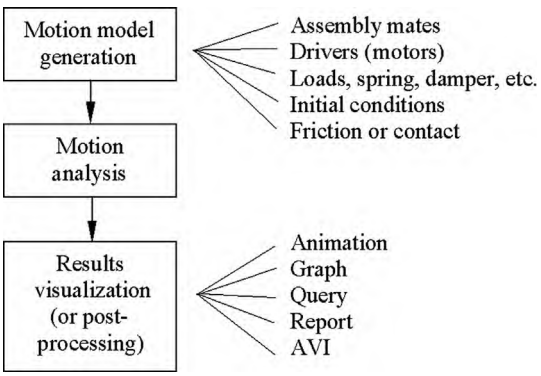


FIGURE 8.12

General process for use of computer tools for motion analysis.

defined in CAD for the mechanism so that the motion model captures essential characteristics and closely resembles the physical behavior of the mechanical system. Note that most motion analysis software accepts assembly mates defined in CAD and converts them into kinematic joints for support of motion analysis.

The analysis or simulation is carried out by a simulation engine, which is a numerical solver that solves the equations of motion for the mechanism. The solver calculates the position, velocity, acceleration, and reaction forces acting on each of the mechanism's moving parts. Typical problems, such as static (equilibrium configuration) and motion (kinematic and dynamic), are supported.

The analysis results can be visualized in various forms. The motion of the mechanism may be animated, or graphs for more specific information, such as the reaction force of a joint in the time domain, may be generated. The user may also query results at specific locations for a given time and may ask for a report on specified results, such as the acceleration of a moving part in the time domain. The motion animation may be saved to an AVI for faster viewing and file portability.

8.4.1 CREATING MOTION MODELS

The basic entities of a valid motion simulation model include ground parts (or ground body), moving parts (or moving bodies), constraints (imposed by assembly mates in CAD), initial conditions (usually the position and velocity of a moving body), and forces and/or drivers. Each of the basic entities is briefly discussed in the following subsections.

8.4.1.1 Ground Parts (or Ground Bodies)

A ground part, or a ground body, represents a fixed reference in space. The first component brought into the CAD assembly is usually stationary, therefore becoming a ground body. Parts (or subassemblies) assembled to the stationary components with no possibility of moving become part of the ground body. Moving and nonmoving parts in the assembly must be identified and assembly mates must be defined to completely constrain the movement of the nonmoving parts.

8.4.1.2 Moving Parts (or Moving Bodies)

A moving part or body represents a single rigid component that moves relative to other parts (or bodies). It may consist of a single part or a subassembly composed of multiple parts. When a subassembly is designated as a moving part, none of its composing parts is allowed to move relative to one another within it.

A moving body has six degrees of freedom—three translational and three rotational—while a ground body has none. That is, a moving body can translate and rotate along the *X*-, *Y*-, and *Z*-axes of a coordinate system. Rotation of a rigid body is measured by referring the orientation of its local coordinate system to the global coordinate system, which is usually the default system of the assembly in CAD. In motion simulation software, the local coordinate system is assigned automatically, usually at the mass center of the part or subassembly. Mass properties, including total mass, moment of inertia, and so forth, are calculated using part geometry and material properties referring to the local coordinate system.

8.4.1.3 Constraints

As mentioned earlier, an unconstrained rigid body in space has six degrees of freedom: three translational and three rotational. When a joint (or a constraint) is added between two rigid bodies, degrees

of freedom between them are removed. In CAD, more commonly employed joints (e.g., revolute, translation, cylindrical) have been replaced by assembly mates. Like joints, assembly mates remove degrees of freedom between parts.

Each independent movement permitted by a constraint (either a joint or a mate) is a free degree of freedom. The free degrees of freedom that a constraint allows can be translational or rotational along the three perpendicular axes. For example, a concentric mate between the propeller assembly and the case of a single-piston engine shown in Figure 8.13 allows one translational DOF (movement along the center axis—in this case the *X*-axis) and one rotational DOF (rotating along *X*-axis). Since the case assembly is stationary, serving as the ground body, the propeller assembly has two free DOF. Adding a coincident mate between the two respective faces of the engine case and the propeller shown in Figure 8.13 removes the remaining translational DOF, yielding a desired assembly that resembles the physical situation—that is, with only the rotational DOF (along the *X*-axis).

In creating a motion model, instead of all movements being completely fixed, certain DOF (translational and/or rotational) are left to allow desired movement. Such a movement is either driven by a motor, resulting in a kinematic analysis, or determined by a force, leading to a dynamic analysis. For example, a rotary motor is created to drive the rotational DOF of the propeller in the engine example. This motor rotates the propeller at a prescribed angular velocity. In addition to a prescribed velocity, the motor may be used to drive a DOF at a prescribed displacement, either translational (using a linear motor) or rotational (using a rotary motor).

It is extremely important to understand assembly mates in order to create successful motion models. In addition to standard mates such as concentric and coincident, CAD (SolidWorks, for example) provides advanced and mechanical mates, as discussed in Chapter 4. Advanced mates provide additional ways to constrain or couple movements between bodies. Mechanical mates

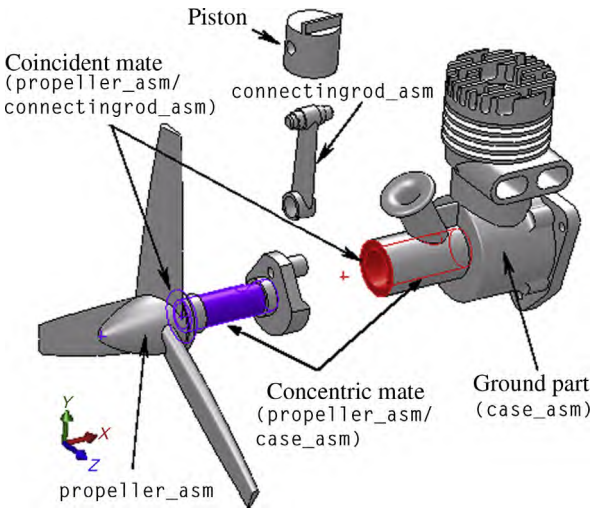


FIGURE 8.13

Assembly constraints defined for the engine model (exploded view).

including cam-follower, gear, hinge, rack and pinion, screw, and universal joint, are essential for motion models yet extremely easy to create in CAD.

In addition to mates, SolidWorks Motion provides 3D contact constraint, which helps to simulate physical problems involving contacts between bodies. Essentially, 3D contact constraint applies a force to separate the parts when they are in contact and prevent them from penetrating each other. The 3D contact constraint becomes active as soon as the parts are in contact.

8.4.1.4 Degrees of Freedom

As mentioned before, an unconstrained body in space has six degrees of freedom: three translational and three rotational. When mates are added to assemble parts, constraints are imposed to restrict the relative motion between them.

Let us go back to the engine example shown in Figure 8.13. A concentric mate between the propeller and the engine case restricts movement on four DOF (*Ty*, *Tz*, *Ry*, and *Rz*) so that only two movements are allowed, one translational (*Tx*) and one rotational (*Rx*). To restrict the translational movement, a coincident mate is added. A coincident mate between two respective faces of the propeller and the engine case restricts movement of three DOF: *Tx*, *Ry*, and *Rz*. Even though combining these two mates achieves the desired rotational motion between the propeller and the case, redundant DOF are imposed: *Ry* and *Rz* in this case.

It is important to understand how to count the overall degrees of freedom for a motion model. For a given motion model, the number of degrees of freedom can be determined using Gruebler's count, defined as

$$D = 6M - N - O \tag{8.105}$$

where

D is Gruebler's count representing the overall degrees of freedom of the mechanism.
M is the number of bodies excluding the ground body.
N is the number of DOF restricted by all mates.
O is the number of motion drivers (motors) defined in the system.

Consider a motion model consisting of the propeller, the engine case, and the rotary motor, in which the propeller is assembled to the engine case by a concentric and a coincident mate (see Figure 8.13). Gruebler's count of the two-body motion model is

$$D = 6 \times 1 - (4 + 3) - 1 = -2$$

However, we know that the propeller can only rotate along the *X*-axis; therefore, there is only 1 DOF for the system (*Rx*), so the count should be 1. After adding the rotary motor, the count becomes 0. The calculation gives us -2 because there are two redundant DOF, *Ry* and *Rz*, which are restrained by both concentric and coincident mates. If we remove the redundant DOF, the count becomes

$$D = 6 \times 1 - (4 + 3 - 2) - 1 = 0$$

Another example is a door assembled to a door frame by two hinge joints. Each hinge joint allows only one rotational movement along the axis of the hinge. The second hinge adds five redundant DOF. Gruebler's count becomes

$$D = 6 \times 1 - 2 \times 5 = -4$$

Again, if we remove the redundant DOF (by removing the second hinge), the count becomes, as it should,

$$D = 6 \times 1 - (2 \times 5 - 5) = 1$$

This is before any motor is added. Usually, redundant constraints are present in a CAD assembly. When a CAD assembly is properly assembled with the desired kinematics, Gruebler's count is often less than 0 because of the redundant DOF constrained.

For kinematic analysis, Gruebler's count must be equal to 0 after adding motors. The solver recognizes and deactivates redundant constraints during analysis. For a kinematic analysis, if you create a model and try to animate it with a Gruebler's count greater than 0, the motion analysis does not run and an error message appears. For the door example, the vertical movement constrained by the second hinge is identified as redundant and removed from the solution. As a result, in a dynamic simulation the entire vertical force is carried by the first hinge. No reaction force is calculated at the second hinge.

To get Gruebler's count to 0, it is often necessary to replace mates that remove a large number of DOF with mates that remove fewer DOF but still restrict the mechanism motion in the same way. Most motion solvers detect redundancies and ignore redundant DOF in all but dynamic analyses. In dynamic analysis, the redundancies can lead to possibly incorrect reaction results, yet the motion is correct. For complete and accurate reaction forces, it is critical to eliminate redundancies from the mechanism. The challenge is to find the mates that impose nonredundant constraints and still allow the intended motion. A combination of a concentric and a coincident mate is kinematically equivalent to a revolute joint, as illustrated in Figure 8.13, between the propeller and the engine case. A revolute joint removes five DOF (with no redundancy); however, combining a concentric and a coincident mate removes seven DOF, among which two are redundant. Using assembly mates to create motion models almost guarantees redundant DOF.

The best strategy is to create an assembly that closely resembles the physical mechanism by using mates that capture the characteristics of the motion revealed in the physical model. That is, an assembly should first be created that correctly captures the mechanism's kinematic behavior. If the purpose of a dynamic analysis is to capture reaction forces at critical components, the assembly mates must be examined to identify redundant DOF. Then, reaction forces at all mates should be checked for the component of interest and only those that make sense should be taken—that is, mostly nonzero forces. Note that zero reaction force is reported at the redundant DOF in most motion simulation software.

8.4.1.5 Forces

Forces are used to operate a mechanism. Physically, forces are produced by motors, springs, dampers, gravity, and so forth. A force entity in motion software can be a force or a torque. Usually motion software provides three types of force: applied forces, flexible connectors, and gravity. Applied forces are those that cause the mechanism to move in certain ways. They are very general, but the force magnitude must be defined by specifying a constant force value or expression function, such as a harmonic function.

Flexible connectors resist motion and are simpler and easier to use than applied forces because only constant coefficients for the forces—a spring constant for example—are supplied. The flexible

connectors include in general translational springs, torsional springs, translational dampers, torsional dampers, and bushings.

A magnitude and a direction must be included for a force definition. A predefined function, such as a harmonic function, may be selected to define the magnitude of the force or moment. For springs and dampers, motion software makes the force magnitude proportional to the distance or velocity between two points, based, respectively, on the spring constant and damping coefficient entered. The direction of a force (or moment) can be defined either along an axis defined by an edge or along the line between two points, where a spring or a damper is defined.

8.4.1.6 Initial Conditions

In motion analysis, initial conditions consist of the initial configuration of the mechanism and the initial velocity of one or more of the mechanism's components. Motion analysis must start with a properly assembled solid model that determines an initial mechanism configuration, composed of position and orientation of individual components. The initial configuration can be completely defined by assembly mates. However, one or more assembly mates must be suppressed, if the assembly is fully constrained, to provide adequate movement.

8.4.1.7 Motion Drivers

Motion drivers (or motors) impose a particular movement on a free DOF over time. A motion driver specifies position, velocity, or acceleration as a function of time, and can control either translational or rotational motion. When properly defined, a motion driver accounts for the remaining DOF of the mechanism that brings Gruebler's count to zero (exactly zero after removing all redundant DOF) or fewer for a kinematic analysis.

8.4.2 MOTION ANALYSIS

The motion solver is capable of solving typical engineering problems, such as static (equilibrium configuration), kinematic, and dynamic. Three common numerical solvers are provided in motion software (e.g., SolidWorks Motion). They are GSTIFF, SI2_GSTIFF, and WSTIFF. GSTIFF is the default integrator and is fast and accurate for displacements. It is used for a wide range of motion simulations. SI2_GSTIFF provides better accuracy of velocities and accelerations, but can be significantly slower. WSTIFF provides better accuracy for special problems, such as discontinuous forces.

Static analysis is used to find the rest position (equilibrium condition) of a mechanism in which none of the bodies are moving. A simple example of static analysis is illustrated in Figure 8.14(a), in which an equilibrium position of a block is to be determined according to its own mass m, the two spring constants k_1 and k_2, and the gravity g. Very often, a static analysis is carried out to find the initial equilibrium configuration of the system before a kinematic or dynamic analysis is conducted.

As discussed earlier, kinematics is the study of motion without regard to the forces that cause it. A mechanism can be driven by a motion driver for a kinematic analysis, where the position, velocity, and acceleration of each link of the mechanism can be analyzed for a given period. Figure 8.14(b) shows a servomotor driving a mechanism at a constant angular velocity. Dynamic analysis is employed for studying the mechanism motion in response to loads, as illustrated in Figure 8.14(c). This is the most complicated and common, and usually more time-consuming, analysis.

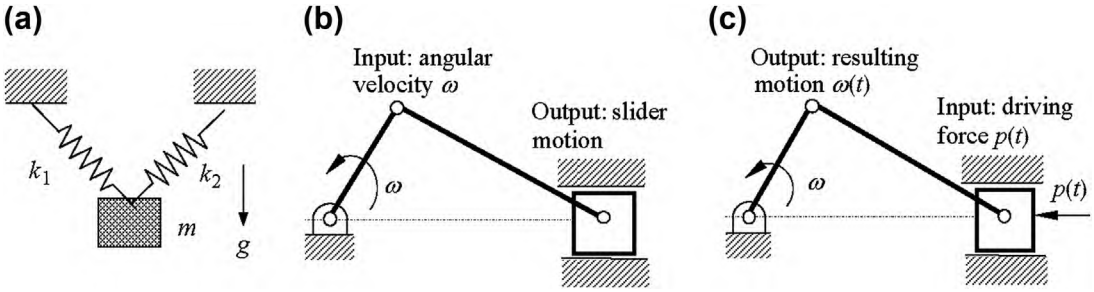


FIGURE 8.14

Common motion analyses: (a) static, (b) kinematic, and (c) dynamic.

8.4.3 RESULTS VISUALIZATION

In motion analysis software, the results of the analysis can be realized using animations, graphs, reports, and queries. Animations show the configuration of the mechanism in consecutive timeframes. They give a global view of the mechanism's behavior, for example, a single-piston engine shown in Figure 8.15(a). The animation may be exported to AVI for other needs.

A joint or a part may be chosen to generate result graphs. An example is the position versus time of the piston in the engine example, as shown in Figure 8.15(b). Graphs provide a quantitative understanding on the characteristics of the mechanism. In addition, most motion simulation software allows checking of interference between bodies during motion. Furthermore, the reaction forces calculated can be used to support finite element analysis of a component using, for example, ANSYS®.

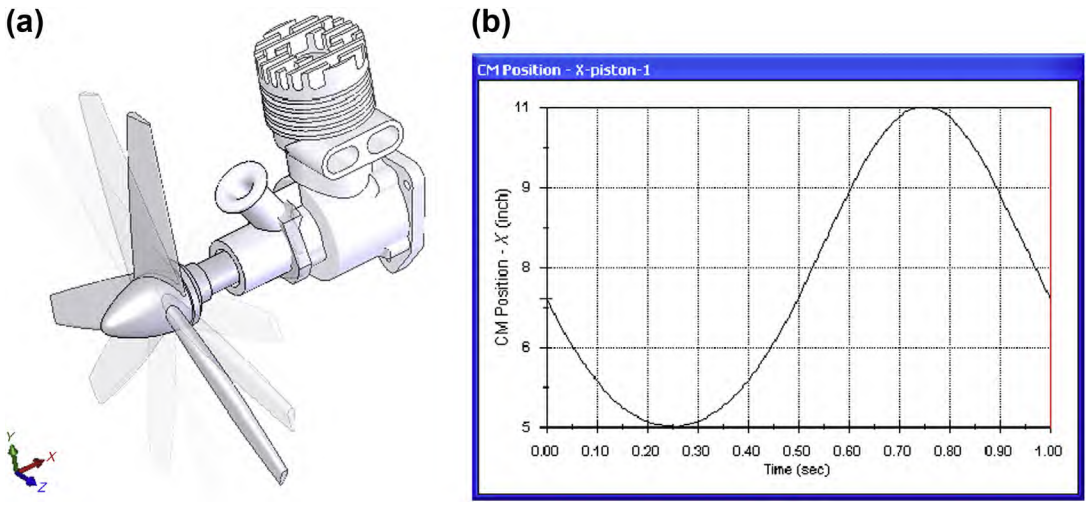


FIGURE 8.15

Simulation result visualization: (a) animation and (b) graph.

8.5 MOTION SIMULATION SOFTWARE

A large number of commercial motion software tools are currently obtainable. Among them are general-purpose codes that support general applications, such as 2D and 3D kinematic and dynamic simulations. Some have strong ties to CAD through designated interfaces; some are even embedded in it. There are also specialized codes, such as CarSim® (www.carsim.com), designed for vehicle dynamic simulations. In this section, a brief overview of commercially available motion codes, including their relative advantages and disadvantages, is presented.

8.5.1 GENERAL-PURPOSE CODES

Adams® and dynamic analysis and design systems (DADS) are the earliest general-purpose codes, becoming available commercially in the late 1970s. They were text based and available only on UNIX systems. Users had to create input data files that defined the mechanical system, including bodies, mass properties, joints, initial conditions, and forces. Analysis results were presented in text files and X-Y graphs. Since the early 1990s, both codes migrated to PCs and incorporated graphics-based user interfaces for pre- and post-processing, which significantly simplified model creation and improved result visualizations. In the early 2000s Adams and DADS were integrated with CAD. For example, LMS released CAT/DADS (integrated with CATIA®, www.lmsintl.com) in 2000, which is now part of the LMS® Virtual.Lab.

Today, these two general-purpose codes remain the most widely used in academia and industry. Both offer capabilities for vehicle dynamic simulations with excellent tire models as well as flexible-body dynamic simulations. These are basically analysis tools for engineers working in dynamic analysis. They are used mainly for support of detail design.

In the 1990s CAD began offering motion simulation capabilities embedded in respective CAD systems, including Pro/ENGINEER Mechanism Design (www.ptc.com), SolidWorks COSMOSMotion™ (renamed SolidWorks Motion after 2008, www.solidworks.com), CATIA Motion (www.3ds.com), NX™ Motion Simulation-RecurDyn (www.plm.automation.siemens.com), and Solid Edge® Motion (www.plm.automation.siemens.com). Very recently, IN-Motion (www.autodesk.com/products/motionbuilder/overview) became a new add-in module for AutoCAD Inventor (www.usaautodesk.com). All of these codes provide a seamless connection from and to their respective CAD systems without the need for any geometric translators. All operations are performed within the CAD environment, including pre-processing, analysis, and post-processing. The learning curve on these software tools is usually relatively flat if the user has CAD experience. However, all of them provide only basic kinematic and dynamic simulation capabilities. CAD-embedded tools are in general less powerful and limited to rigid bodies. They are usually more error-prone but much easier to learn because of their simplicity and CAD connections. They are more or less designers' tools, mainly for support of mostly conceptual design.

8.5.2 SPECIALIZED CODES

Specialized codes usually offer capabilities in specific engineering fields in addition to standard capabilities. Two popular specialized codes are commercially available: CarSim (Mechanical Simulation Corp.) and Adams/Car (MSC Software Corp., www.mscsoftware.com). Both support vehicle dynamics simulations.

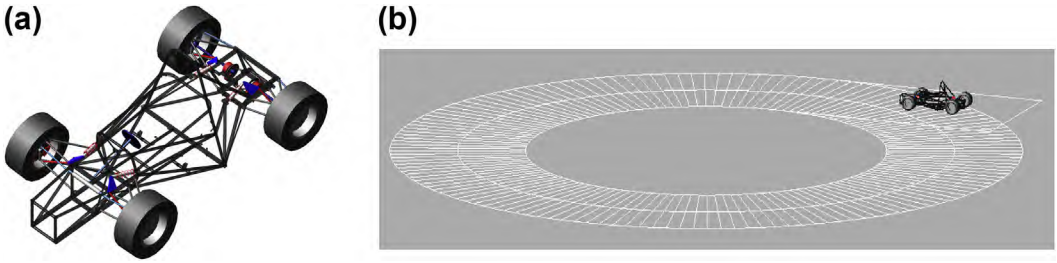


FIGURE 8.16

Vehicle dynamic simulation of a Formula SAE racecar: (a) 15-DOF Adams/Car model, and (b) skid pad racing (constant-radius cornering simulation).

The Windows-based CarSim version was first released in 1997. CarSim simulates vehicle dynamics, where the vehicle model consists of braking, accelerating, handling, and riding. CarSim's vehicle model can respond to driver control, ground, and aerodynamics. This software supports scenarios such as rollover of double lane changes and stability of trailer towing.

Adams/Car is a template-based modeling and simulation tool that helps journeyman engineers (especially students) speed up and simplify the vehicle modeling process. With Adams/Car, users can simply enter vehicle model data into the templates and the program automatically constructs subsystem models (e.g., engine, shock absorbers, tires) as well as full vehicle assemblies. Once these templates are created, they can be made available to novice users of the software, enabling them to perform standardized vehicle maneuvers.

A Formula SAE racecar model developed by engineering students was converted into an Adams/Car model using the provided templates, as shown in Figure 8.16(a). The vehicle model was then simulated for skid pad racing, a constant-radius cornering simulation shown in Figure 8.16(b).

8.6 CASE STUDIES

In this section, four case studies and two tutorial examples are presented. The case studies involve a broad range of applications, including a kinematic study of a racecar suspension, the design of a HMMWV suspension, driving simulators, and recreational waterslides. The purpose of these case studies is to demonstrate the engineering capabilities of motion analysis software and some of its common industry applications. Tutorial examples, including a sliding block mechanism and a single-piston engine, are also presented. Step-by-step instructions for creating these tutorial examples are given in Project S2 and P2. Model files are available for download on this book's companion website (http://booksite.elsevier.com/9780123820389).

8.6.1 FORMULA SAE RACECAR

The Formula SAE (Society of Automotive Engineers) racecar study involves kinematic and dynamic analyses of a racecar-style suspension, as shown in Figure 8.17. Each year engineering students throughout the world design and build formula-style racecars and participate in annual Formula SAE competitions (students.sae.org). The competition is a meaningful engineering experience that provides an opportunity for students to work in a dedicated team environment.

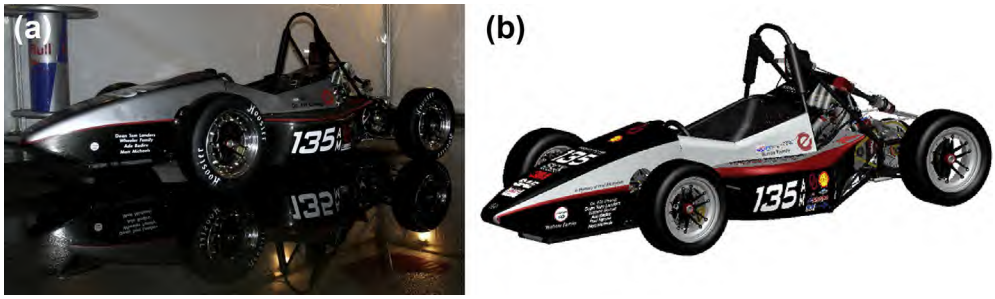


FIGURE 8.17

Formula SAE racecar designed and built by engineering students: (a) physical racecar and (b) virtual racecar designed in Pro/ENGINEER.

The suspension of the entire racecar was modeled for both kinematic and dynamic analyses. The analysis results were validated using experimental data, acquired by mounting a data acquisition system on the racecar and driving the car on the test track following specific driving scenarios that were consistent with those of the simulations. The results were used to aid the suspension design for handling and cornering (Wheeler, 2006). Assembling an entire vehicle suspension for motion analysis is nontrivial and beyond the scope of this book. Therefore, only the right front quarter of the suspension, as shown in Figure 8.18, is included in this section.

The purpose of this case study is mainly to highlight capabilities in Pro/ENGINEER Mechanism Design and SolidWorks Motion for supporting design of the kinematic characteristics of vehicle suspension. The motion model was first created in Pro/ENGINEER for kinematic analysis, and then imported into SolidWorks for dynamic analysis and design studies. The road profile is characterized by the geometric shape of a profile cam, which is assembled to the tire using a cam-follower connection.

The quarter suspension consists of major components that essentially define the kinematic and dynamic characteristics of the racecar. These components include upper and lower control arms,

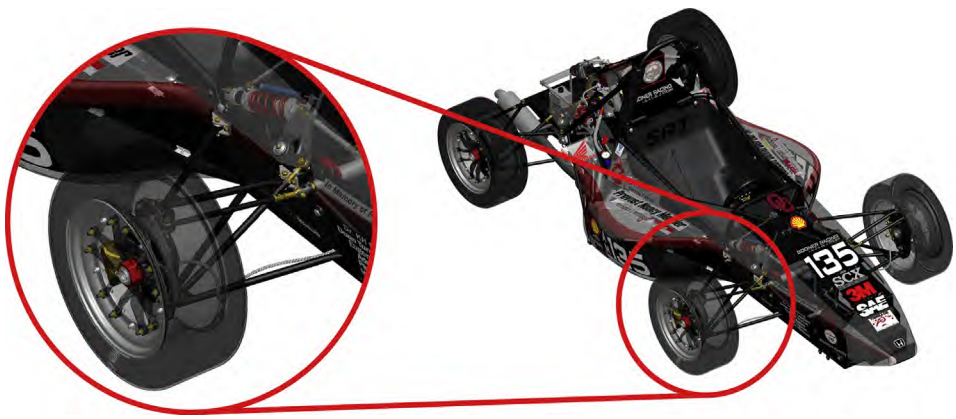


FIGURE 8.18

Right front quarter of the racecar suspension.

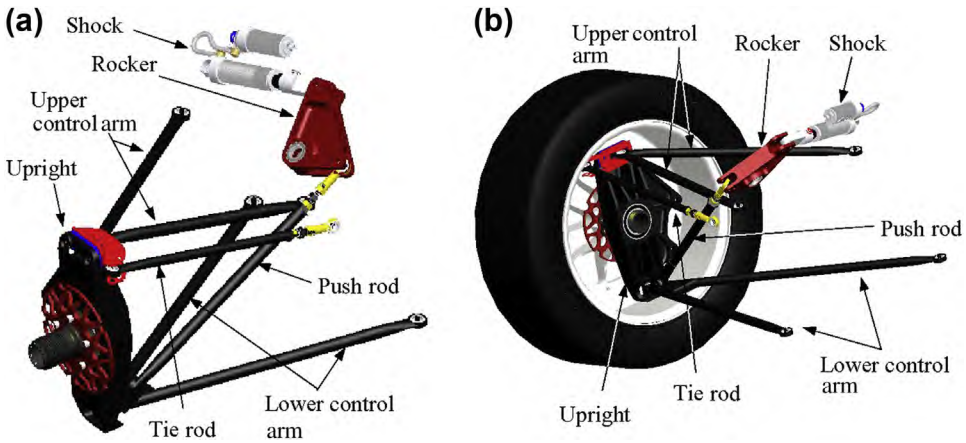


FIGURE 8.19

Major components of the quarter suspension: (a) saved view (View A) and (b) saved view (View B).

upright, rocker, shock, push rod, tie rod, and wheel and tire, as shown in Figure 8.19. The dangling end of the shock, both control arms, rocker, and tie rod are connected to the chassis frame using numerous joints. The chassis frame was assumed fixed and the tire is pushed and pulled by the profile cam (not shown), mimicking the road profile. Two views, shown in Figure 8.19, were created to aid the visualization of the assembled model.

The tire of the quarter suspension is in contact with the profile cam that characterizes the road profile. As shown in Figure 8.20(a), the geometry of the cam consists of two circular arcs of radius 7.65 in. (AB and FG), which are concentric with the cam center. Therefore, when the cam rotates, these two circular arcs do not push or pull the tire; the result is two flat segments of the road profile, as shown in Figure 8.20(b). In addition, the circular arc CDE is centered 4 in. above the cam center with a radius of 4 in. When the cam rotates, arc CDE pushes the tire up, mimicking a hump of 1.35 in. (that is $8 - 7.65 = 1.35$ in., peak at point D). A ditch is characterized by an 8 in. arc (HIJ) centered at 3 in. above the cam center. As the cam rotates, arc HIJ creates a ditch 1.65 in. deep (that is $7.65 - (8 - 3) = 1.65$ in.). The remaining straight lines and arcs provide smooth transitions between flats, humps, and ditches in the road profile.

Based on the geometry of the profile cam, this quarter suspension goes over a 1.35 in. hump and a 1.65 in. ditch in one complete rotation of the cam. Note that since the radius of arc AB is 7.65 in. the cam causes the quarter suspension to travel roughly 41.8 in. (3.48 ft) in one complete rotation. Since the profile cam rotates two complete cycles in one second, the suspension travels about 6.96 ft/sec (i.e., 4.74 MPH), which is very slow.

There are nine bodies in this motion model, including the ground body. There are two rigid (no symbol) joints, three pin joints, eight ball joints, and one cylinder joint, as shown in Figure 8.21. A servomotor that rotates the profile cam for 1 sec was added to conduct a kinematic analysis. Several measures are critical in determining the pros and cons of the suspension design. Among them, the most important one is probably the camber angle. We will show the camber angle results momentarily. Note that the camber angle is defined as the rotation of the upright along the *X*-axis of *WCS* (World Coordinate System, which is the reference frame of the motion model). First, we look at the shock travel distance.

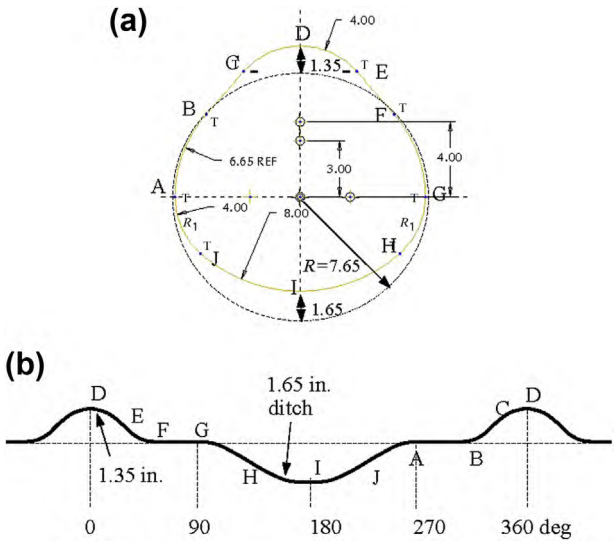


FIGURE 8.20

Road profile: (a) geometry of the profile cam and (b) road profile generated by the profile cam.

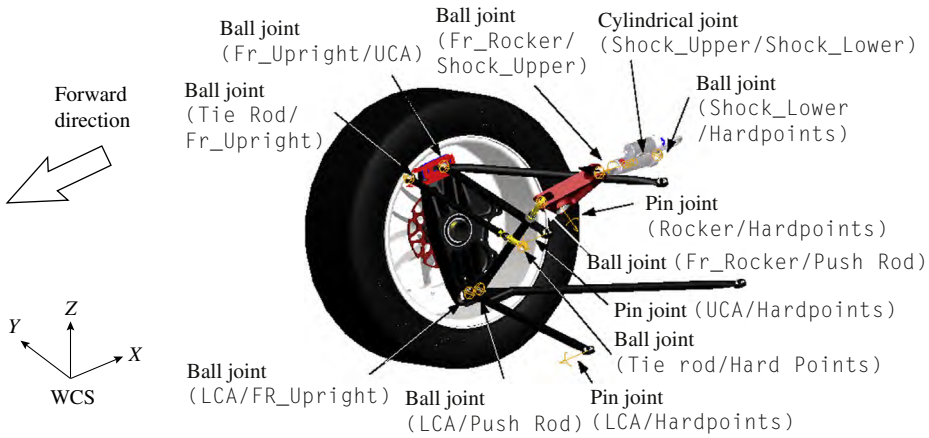


FIGURE 8.21

Joints defined for the quarter suspension assembly. UCA = upper control arm; LCA = lower control arm.

The graph in Figure 8.22(a) shows that the shock travels between about 6 in. and 8.5 in. The overall travel distance is about 2.5 in., which is probably too large for such a small hump or ditch. In fact, in the simulation it appeared that the shock was compressed too much, allowing the piston to penetrate its reserve cylinder. In reality, this would not happen; however, the simulation raised a flag indicating that there could be severe contact within the shock, leading to potential part failure.

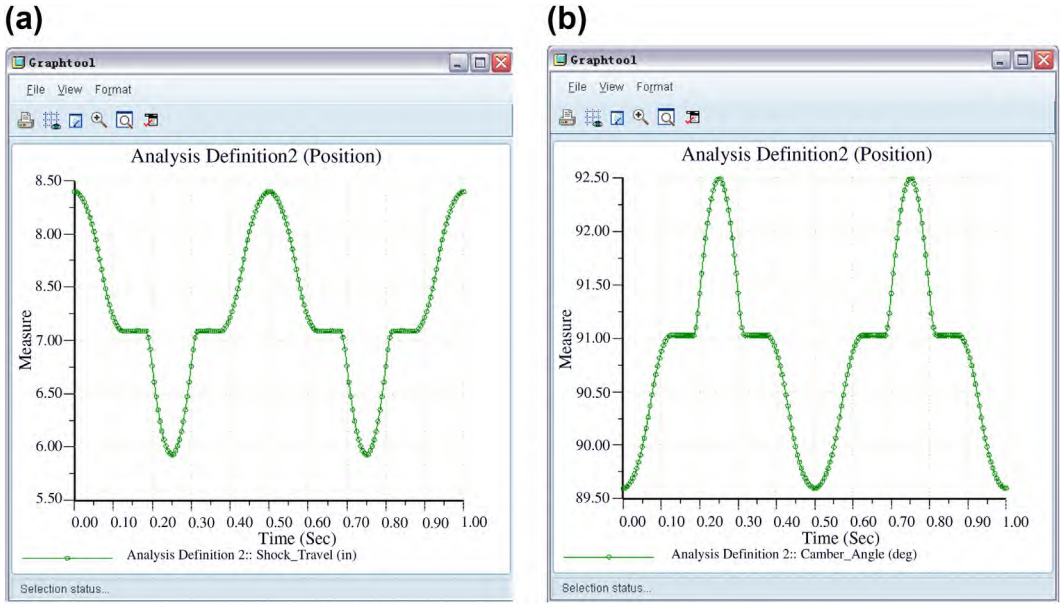


FIGURE 8.22

Result graphs: (a) shock travel and (b) camber angle.

The camber angle is made by the wheel of an automobile—specifically, between the vertical axis of the wheel and the vertical axis of the vehicle when viewed from the front or rear. It is used in the design of steering and suspension. If the top of the wheel is further out than the bottom (that is, away from the axle), it is called positive camber; if the bottom of the wheel is further out than the top, it is called negative camber. In this model, the camber angle is defined as the rotation angle of the upright along the *X*-axis of *WCS*.

As shown in Figure 8.22(b), the camber angle was set to about 91 (or −1) deg. on the flat terrain. The camber angle varied to 92.5 (or −2.5) and 89.5 (or 0.5) deg. respectively, when the tire went over the hump and the ditch. In general, camber angle alters the handling quality of a particular suspension design; in particular, negative camber improves grip when cornering because it places the tire at a more optimal angle to the road, transmitting the forces through the tire's vertical plane rather than through a shear force across it. However, excessive negative camber change in the hump can cause early lockup under braking or wheel spin under acceleration.

The Pro/ENGINEER model of the quarter suspension was imported into SolidWorks and a motion model was constructed with all assembly mates defined according to the Pro/ENGINEER motion joints. A guide cylinder was used to define the cam mate in SolidWorks. The motion study was carried out in SolidWorks Motion, where three measures, consistent with those defined in the Pro/ENGINEER kinematic analysis, were recorded, including vertical wheel travel, shock travel, and camber angle (Figure 8.23). The result graphs show that the kinematic analysis using SolidWorks Motion yielded identical results compared with the Pro/ENGINEER analysis. These results indicate the accuracy of the model translation and reassembly and imply that the quarter suspension kinematic analysis can be duplicated in SolidWorks.

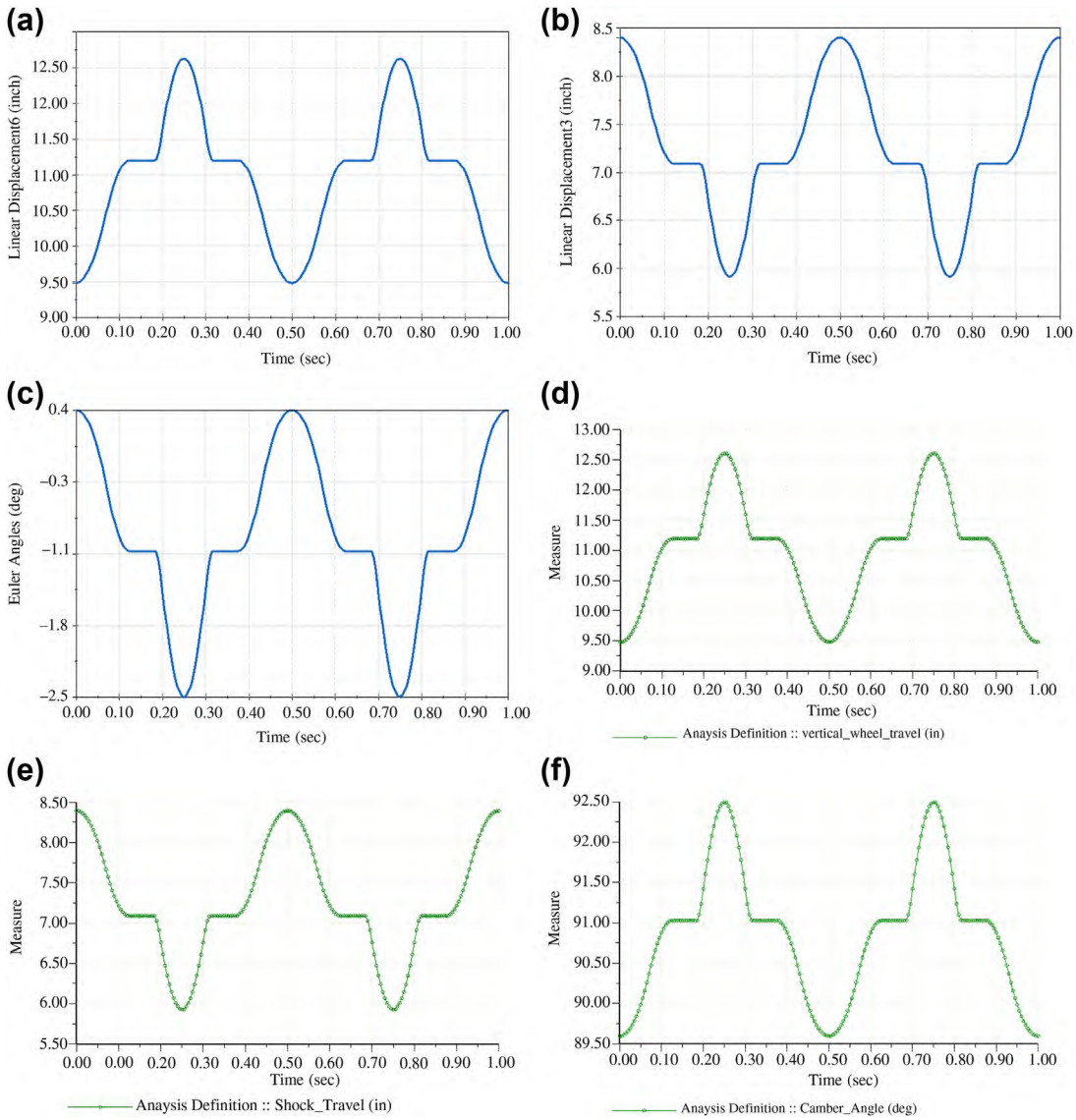


FIGURE 8.23

Verification of kinematic analysis results: (a) SolidWorks vertical wheel travel; (b) SolidWorks shock travel; (c) SolidWorks camber angle; (d) Pro/ENGINEER vertical wheel travel; (e) Pro/ENGINEER shock travel; (f) Pro/ENGINEER camber angle.

FIGURE 8.24

External force of 150 lb applied to the tire.

The dynamic analysis of the quarter suspension was performed by taking racecar weight, spring rate, and shock damping into consideration. As shown in Figure 8.24, a 150 lb external force pointing upward was applied on the road profile cam to mimic the wheel load due to racecar weight (445 lb) and driver weight (155 lb). An equilibrium analysis was first carried out. The equilibrium state of the racecar was assumed as the initial condition for the dynamic simulation, in which the racecar started in equilibrium on the flat road and then reached the first hump.

A spring and a damper were also defined in the dynamic analysis, as shown in Figure 8.25. The physical position of the spring is shown in Figure 8.26. The spring rate and the damping coefficient were 100 lb/in. and 10 lb/(in./sec), respectively. The free length of the spring was 5.5 in. Note that when the shock was fully extended to its maximum length, the spring length was 4 in., which implies a 150 lb preload.

The dynamic simulation (Case A) was carried out assuming a racecar speed of 4.74 mph. The shock travel is shown in Figure 8.27. Note that the shock length was allowed to vary between 7.3 in. and 9 in., as shown in Figure 8.28. This means the shock travel obtained in this dynamic simulation (Case A) is acceptable and the design is safe.

Another scenario (Case B) was created where a modified profile cam with a larger hump (4.35 in.) was used, as shown in Figure 8.29. To avoid resonance, a segment velocity (Figure 8.30) was assigned to ensure adequate time for the suspension to return to the equilibrium state between each hump. As can be seen from the resultant shock travel graphs (Figures 8.31(a) and (b)), in Case B the shock was compressed too much and the shock travel exceeded the permitted range, which might have led to a part failure. Making design changes therefore became necessary to bring the shock travel back to the safe range.

In this study, the three design variables investigated were the spring preload, the shape of the rocker, and the length of the push rod. The preload of the spring was 150 lb at the current design, which

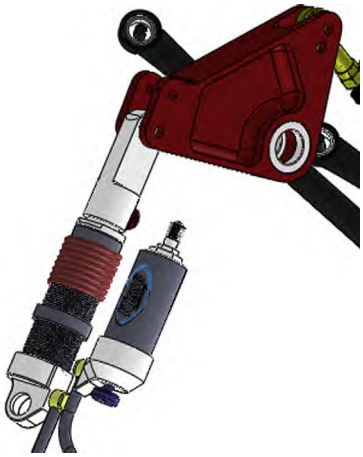


FIGURE 8.25

Spring and damper in the suspension.

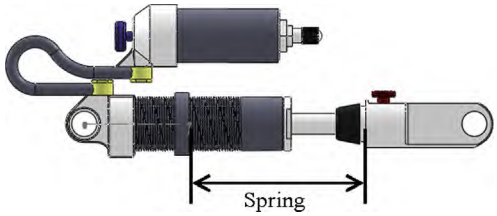


FIGURE 8.26

Physical position of the spring.

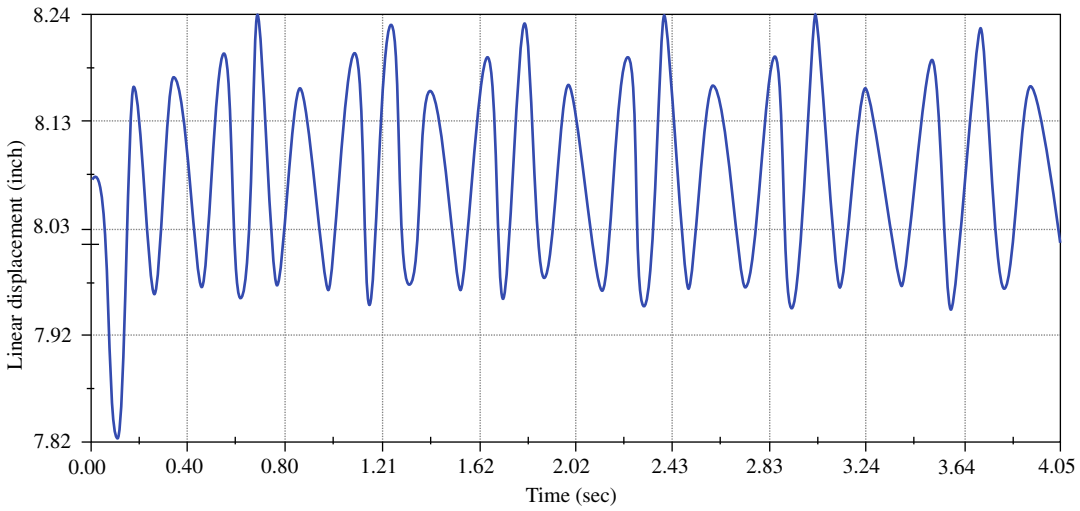


FIGURE 8.27

Shock travel (Case A).

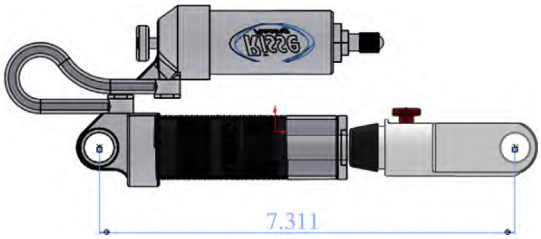


FIGURE 8.28

Shock travel allowed.

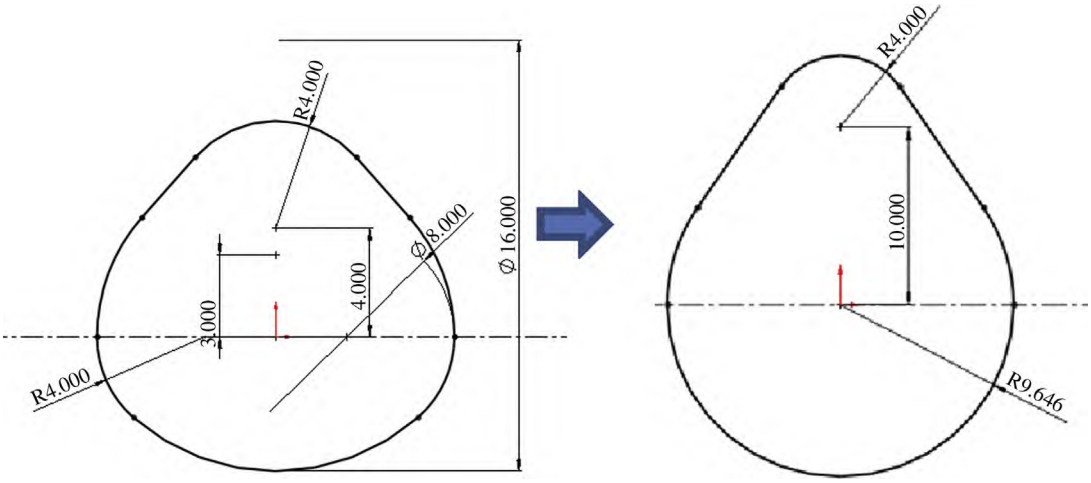


FIGURE 8.29

Modified profile cam with a larger hump.

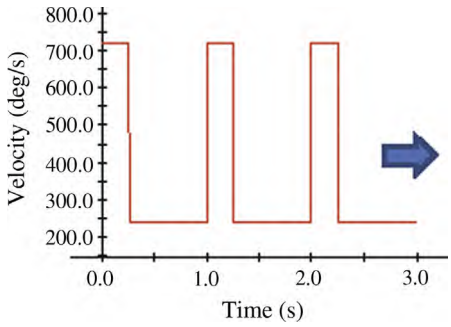


FIGURE 8.30

Segment velocity added for Case B.

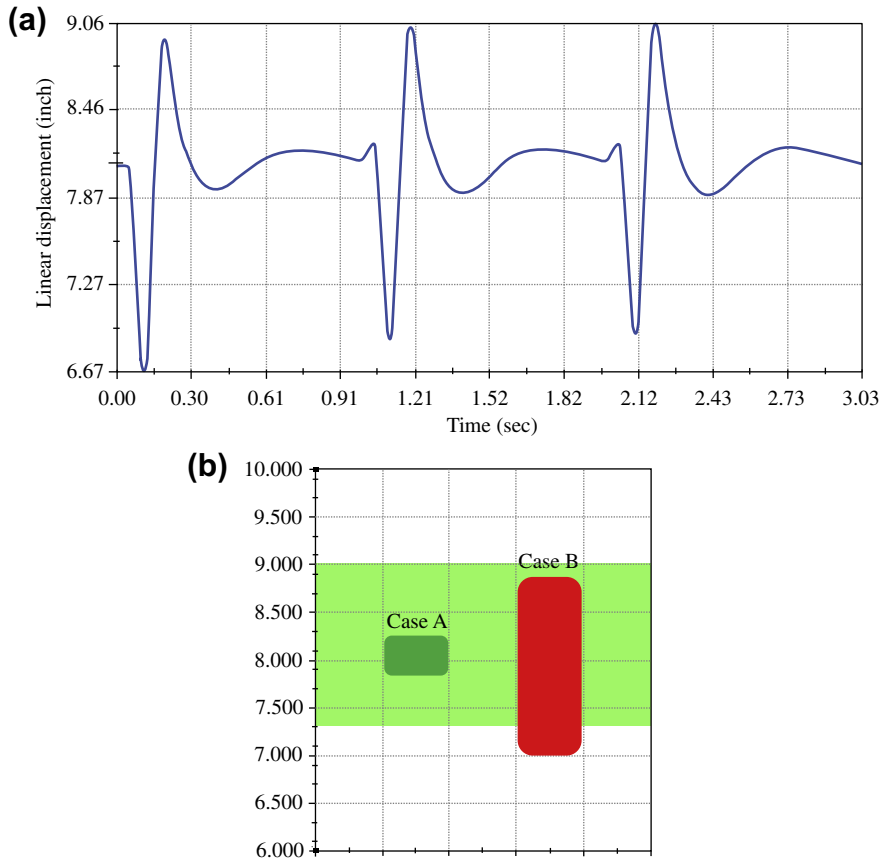


FIGURE 8.31

Shock travel: (a) Case B in the time domain and (b) comparison of Cases A and B.

means that the spring was compressed by 1.5 in. when the shock was fully extended. The spring preload was increased from 150 lb (current design) to 200 lb and 250 lb; the resultant shock travels are shown in Figure 8.32. As can be seen, as the preload increased, the overall shock length slightly decreased but the piston moved further away from the cylinder. This is because as long as the compressive force on the shock is larger than the preload, the shock with a relatively large preload will be less compressed than one with a small preload under the action of the same force.

The rocker component was parameterized in SolidWorks so that its shape could be adjusted. When the rocker shape changed (Figure 8.33(a)), the overall shock length could be reduced while the piston moved closer to the cylinder (Figure 8.33(b)). The length of the push rod did not have much capacity for change (less than 1 in.). It also turned out that varying the push rod length led to only a very small change in the shock travel. As can be seen from Figure 8.31(b), to ensure a safe shock travel, the overall shock length needs to be reduced while the piston should move further away from the cylinder.

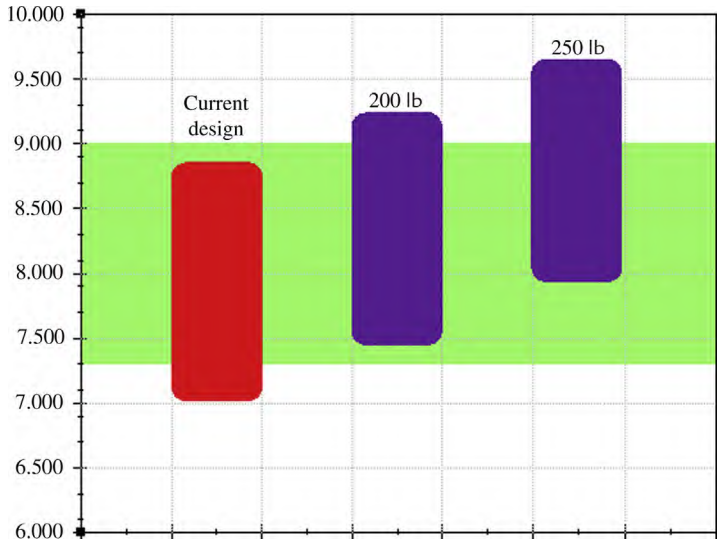


FIGURE 8.32

Resultant shock travel distances for the current design (150 lb preload), 200 lb and 250 lb preload.

A feasible design was then achieved by simultaneously increasing the spring preload to 250 lb and changing the rocker height to 2.7 in. The resulting shock travel is shown in Figure 8.34.

It was also noticed that the wheel camber angle at the current design might not meet the racecar requirement; that is, a negative camber angle is more desirable. The camber angle of the wheel could be adjusted by adding and removing slims from between the toe link and upright. To ensure a negative camber angle at any time for the road profile in Case B, some of the slims needed to be removed from the suspension. Therefore, the thickness of the slim block, which is an equivalent component substituted for the slims, was reduced so that only one thin slim was left in the system, as shown in Figure 8.35(a). The resultant camber angle change through time is graphed in Figure 8.35(b). As can be seen, the camber angle became negative at all times. In addition, it was observed that changing the camber angle would have almost no effect on the shock travel. Therefore, the design was still satisfactory after the camber angle had been adjusted.

The Formula SAE racecar case study illustrates the numerous motion analyses and designs that can be carried out using Pro/ENGINEER and SolidWorks. However, although much useful information was obtained by the kinematic and dynamic analyses using the quarter suspension model, ultimately a full-vehicle dynamic simulation must be carried out to fully understand the suspension design and, hopefully, to develop a strategy for design improvement.

8.6.2 HIGH-MOBILITY MULTIPURPOSE WHEELED VEHICLE

The high-mobility multipurpose wheeled vehicle (HMMWV) example discussed in Chapter 1 is presented here, in more detail, to illustrate dynamic simulation and design. More than 200 parts and assemblies were created in the CAD model, as shown in Chapter 5, Figure 5.23(a). The suspension was modeled in detail (Chapter 5, Figure 5.23(b)) such that the main aspects of the vehicle's performance

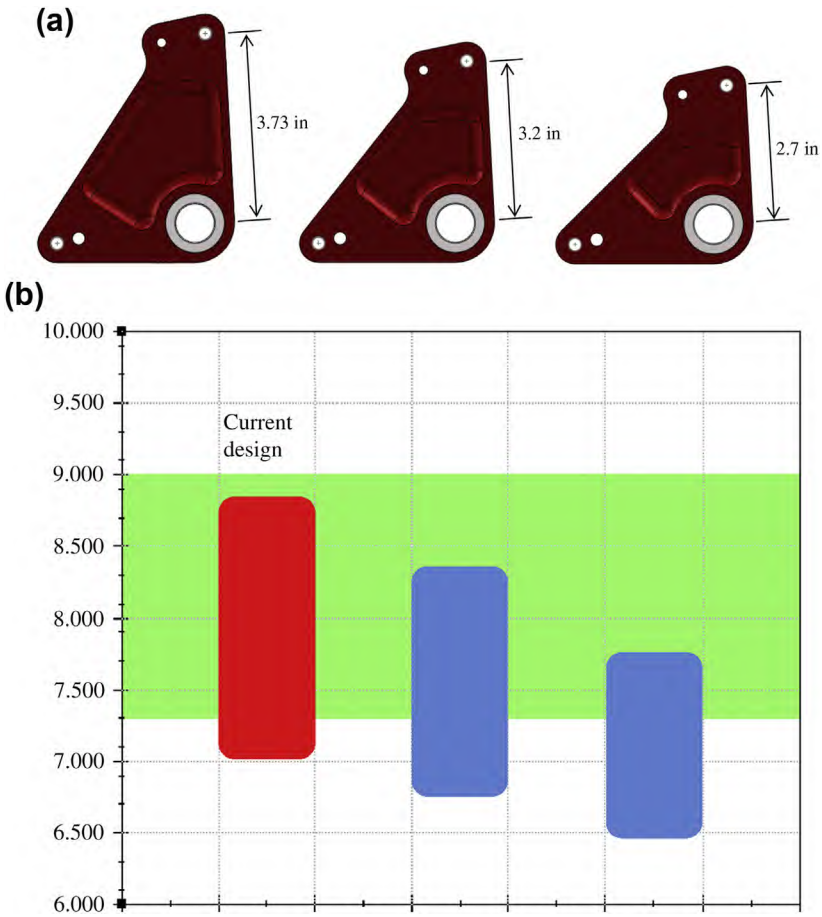


FIGURE 8.33

(a) Rocker shape change and (b) impact of the rocker shape change on the overall shock travel.

could be captured accurately in motion simulation. A more detailed view of the front right suspension quarter is shown in Figure 8.36(a). A dynamic simulation model of 18 bodies and 21 joints, shown in Figure 8.36(b), was created and simulated in DADS with a total of 17 sec and a time step of 0.001 sec (Chang and Joo 2006).

A 100 ft $\times$ 100 ft terrain was used for simulation (see Chapter 5, Figure 5.24(b)). Note that the terrain was fairly bumpy. The maximum height of the bumps on the terrain was 7.68 in. The vehicle vibrated significantly toward the later stage of the simulation because of bumpy road conditions. In this model, the vehicle was "driven" by a constant angular velocity of 1.53 rev/sec applied at the four wheels, which produced a path that went through both bumpy areas. In this case, the vehicle was moving at a constant speed of 9.54 mph.

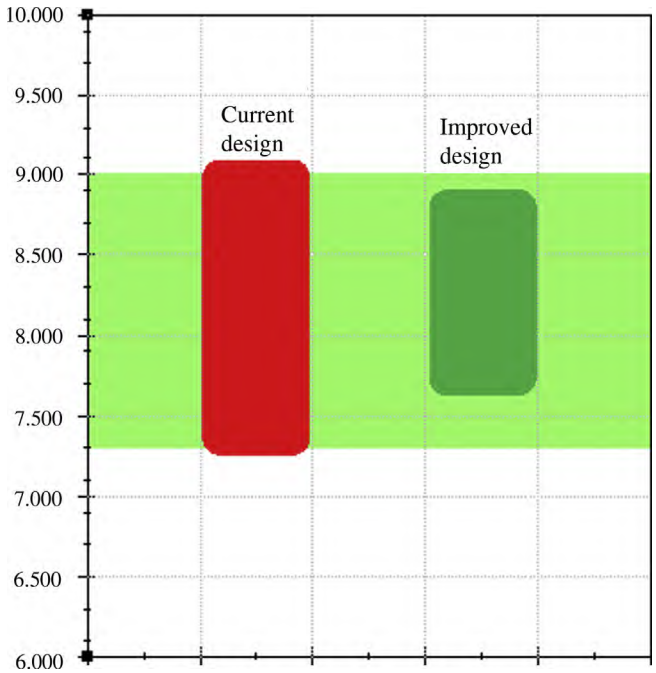


FIGURE 8.34

Shock travel of the improved design by setting spring preload to 250 lb and changing the rocker height to 2.7 in.

The design problem was defined as follows:

$$\begin{aligned}
&\text{Minimize:} && \phi(\boldsymbol{b}) = \frac{1}{T}\int_0^T [F(\mathbf{p}(t))]^2 dt \\
&\text{Subject to:} && \psi_\alpha(\boldsymbol{b}) = (z_{w_i}(t) - 18) - h_i(t) \le \psi_\alpha^u, \quad \alpha = 1,4 \\
& && \psi_\beta(\boldsymbol{b}) = -[(z_{w_i}(t) - 18) - h_i(t)] \le \psi_\beta^u, \quad \beta = 1,4 \\
& && \psi_9(\boldsymbol{b}) = |\ddot{z}_{ds}(t)| \le \psi_9^u \\
& && \psi_{10}(\boldsymbol{b}) = |z_{ds}(t) - z_{ch}(t)| \le \psi_{10}^u \\
& && \psi_\gamma(\boldsymbol{b}) = |z_{w_i}(t) - z_{ch}(t)| \le \psi_\gamma^u, \quad \gamma = 1,4 \\
& && \boldsymbol{b}_j^\ell \le \boldsymbol{b}_j \le \boldsymbol{b}_j^u, \quad j = 1,3
\end{aligned} \tag{8.106}$$

where

$\phi(\boldsymbol{b})$ represents energy absorption ability of the vehicle suspension at the driver's seat.
$z_{wi}(t)$ is the z-coordinate of the ith wheel center.

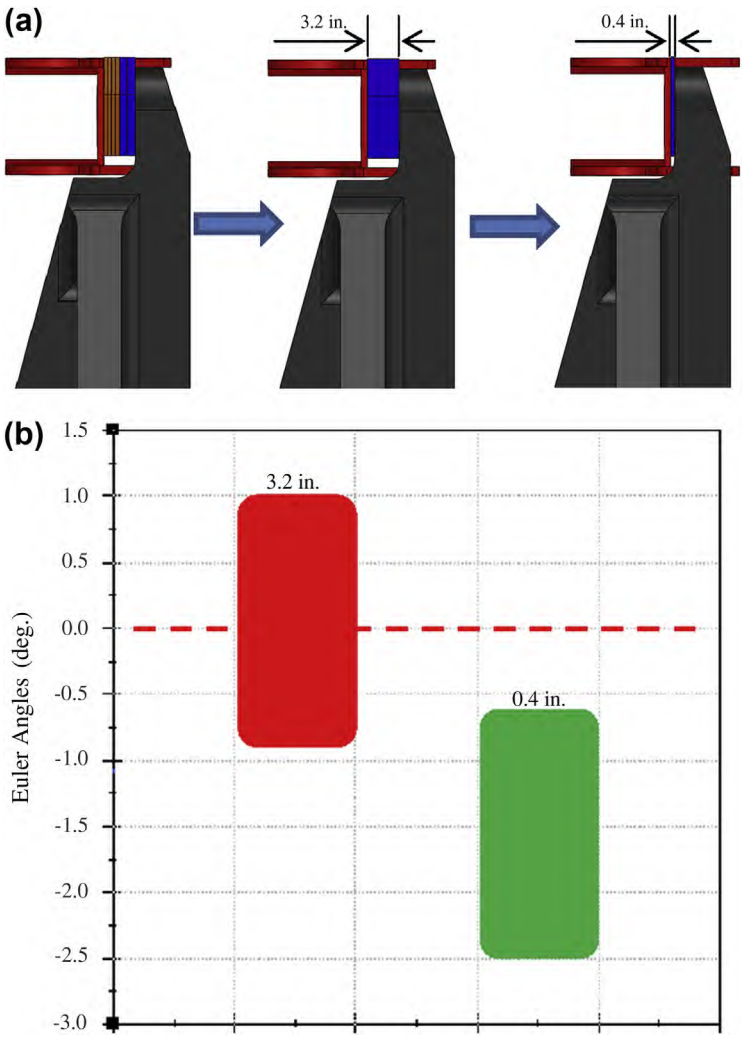


FIGURE 8.35

Impact of adding slims to the camber angle.

$h_i(t)$ is the height of the road profile corresponding to the ith wheel at the given time t.
$z_{ds}(t)$ and $\ddot{z}_{ds}(t)$ are the driver seat position and acceleration, respectively, in the z-direction of the global coordinate system (vertical).
$z_{ch}(t)$ is the z-displacement of the chassis with respect to the global coordinate system (shown in Figure 8.36(b)).

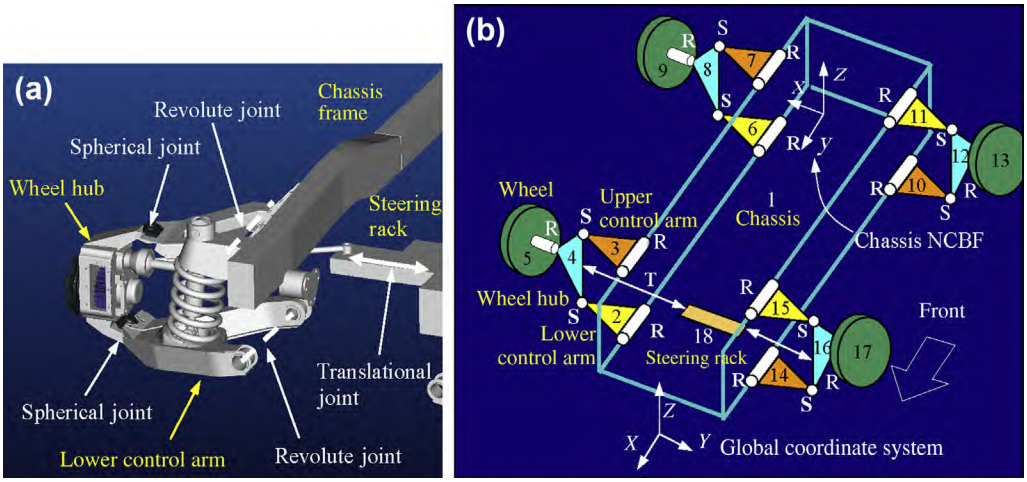


FIGURE 8.36

HMMWV dynamic simulation: (a) CAD model of the front suspension and (b) schematic view of the simulation model.

Note that in Eq. 8.106, $\psi_\alpha(\boldsymbol{b})$ and $\psi_\beta(\boldsymbol{b})$ essentially characterize the deformation of the tires. ψ^u simply represents the upper bound of the respective constraints $\psi(\boldsymbol{b})$. Note also that the tire radius is 18 in. and $z_{wi}(t) - 18$ is 0 if no deformation occurs in the tire. $\psi_\gamma(\boldsymbol{b})$ specifies the wheel center position with respect to the chassis in the vertical direction.

The function in the integrand $F(\boldsymbol{p}(t))$ of Eq. 8.106 is defined as follows (U.S. Tank-Automotive Research and Development Command, 1979),

$$F(\boldsymbol{p}(t)) = p_1(t) - 0.108p_4(t) + 0.25p_6(t) - p_7(t) \tag{8.107}$$

where $p_i(t)$ can be computed from the absorbed power equations,

$$\begin{bmatrix} \dot{p}_1(t) \\ \dot{p}_2(t) \\ \dot{p}_3(t) \\ \dot{p}_4(t) \\ \dot{p}_5(t) \\ \dot{p}_6(t) \\ \dot{p}_7(t) \end{bmatrix} = \begin{bmatrix} -29.8p_1(t) - 497.49\ddot{z}_s(t) - 100.0p_2(t) \\ 10.0p_1(t) \\ 1736.9p_1(t) - 108.0p_4(t) \\ 100.0p_1(t) - 35.19p_3(t) - 39.1p_4(t) \\ -315.7p_1(t) + 34.0956p_4(t) + 171.075p_6(t) \\ -80.0p_1(t) - 91.36p_4(t) - 30.28p_5(t) \\ p_1(t) - 0.108p_4(t) + 0.25p_6(t) - 6.0p_7(t) \end{bmatrix} \tag{8.108}$$

for $0 \le t \le T$, with initial conditions

$$p_i(0) = 0, i = 1, 7 \tag{8.109}$$

The initial conditions were reset at each time step during numerical calculation.

Table 8.1 Upper Bounds of Constraint Functions

Performance Function	Description	Upper Bound
ψ_α^u	Jounce of each wheel	1.25 in.
ψ_β^u	Rebound of each wheel	3.5039 in.
ψ_9^u	Driver's seat acceleration	0.75 g
ψ_{10}^u	Driver's seat position w.r.t. chassis	3.5 in.
ψ_γ^u	Wheel center position w.r.t. chassis	12.0 in.

Three design variables were defined for the HMMWV: vehicle track, wheelbase (see Figure 5.25), and percentage change in thickness of the lower control arm. Constraint function bounds are shown in Table 8.1. The constraint functions were evaluated at every 0.01 seconds during the total 17-second simulation period, except for the z-accelerations at the driver seat, where the step size was refined to be 0.001 seconds since the z-accelerations directly contributed to the objective function. Therefore, there were 39,100 ($13 \times 100 \times 17 + 1 \times 1000 \times 17$) constraint functions to process at each design iteration. Note that most of the constraint function values at the initial design were less than their respective upper bounds, except for a few time steps of driver seat acceleration $\psi_9(\boldsymbol{b})$ and driver seat position $\psi_{10}(\boldsymbol{b})$ (see Figures 8.37(a) and (b), respectively). This meant that the initial design was infeasible.

The optimization took 18 iterations to converge using the modified feasible direction (MFD) method. At the optimal design the objective function was reduced by 31.3%, and all performance constraints were satisfied. The track (b1) and wheelbase (b2) design variables increased by 17.6% and 14.6%, respectively. The percentage thickness of the lower control arm (design variable b3) decreased by 17.9%, as summarized in Table 8.2.

Note that the reduction in objective function value was due to the significant decreasing z-acceleration values at the driver seat (Figure 8.37(a)). Also, the distance between the driver seat and the chassis in the z-direction was significantly reduced (Figure 8.37(b)). The rest constraint functions also indicated that the vehicle became smoother while moving along the same paths. This was due to the fact that both vehicle track and wheelbase were increased, which contributed to a wider and longer chassis and therefore more stability and less vibration. The change in HMMWV suspension geometry is shown in Chapter 5, Figure 5.29.

8.6.3 DRIVING SIMULATORS

Driving simulators are probably the most sophisticated application of computer-aided kinematic and dynamic simulations, as well as one of the biggest triumphs in their development. Similar to flight simulators, driving simulators place the driver in an artificial environment believed to be a valid substitute for one or more aspects of the actual driving experience. However, unlike flight simulators developed mainly for pilot training, driving simulators support much more than driver training. Advanced driving simulators today are used by engineers and researchers in vehicle design, intelligent highway design, and human factors studies such as driver behaviors under the influence of drugs, alcohol, and severe weather conditions. They provide a safe environment for testing in which controlled, repeated measurements can be undertaken cost-effectively. Researchers and engineers

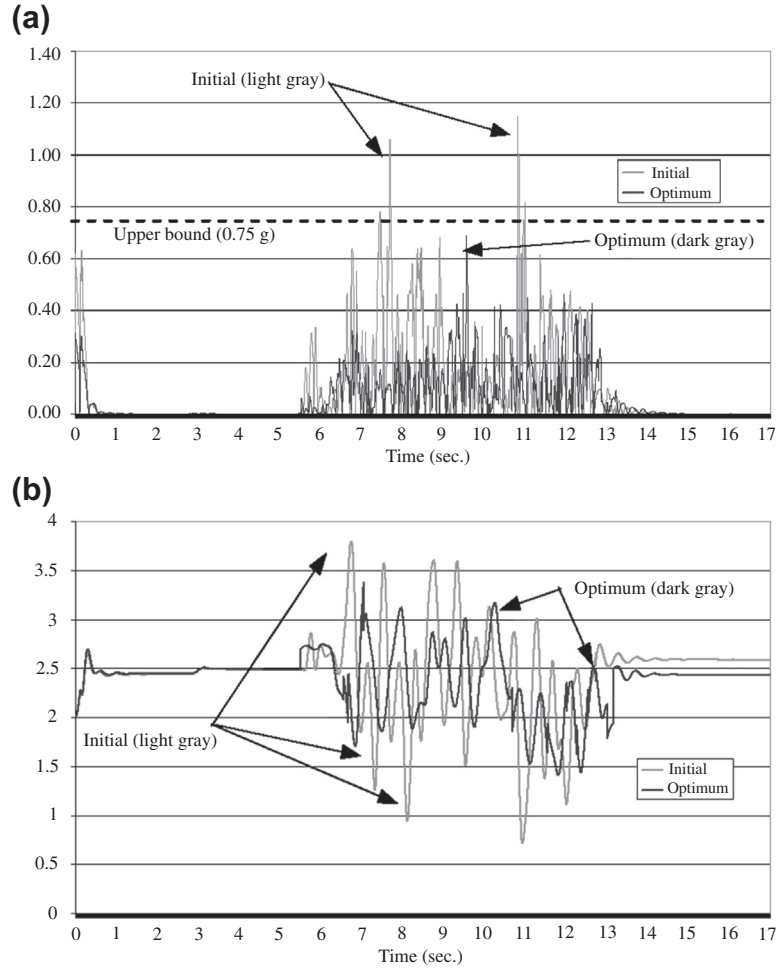


FIGURE 8.37

Change of vehicle performance: (a) driver's seat accelerations $\psi_9(\boldsymbol{b})$ (g), and (b) driver seat position $\psi_{10}(\boldsymbol{b})$ (in.).

Table 8.2 Design Optimization of the HMMWV Dynamic Model

Measure	Initial Design	Optimal Design	% Change
$\phi(\boldsymbol{b})$	0.00595 W	0.00405 W	−31.3
b1	21.68 in.	25.52 in.	+17.6
b2	50.69 in.	58.12 in.	+14.6
b3	100% original thickness	82.0% original thickness	−17.9

believe that the measurements obtained can help them predict equivalent measurements in the real world that lead to a better understanding of the complex driver–vehicle–roadway interaction in critical driving situations. The results of such studies will ultimately lead to reductions in the number of traffic-related deaths and injuries on the nation's highways.

From its appearance, a driving simulator—for example, the University of Iowa's National Advanced Driving Simulator (NADS) in Iowa City—consists of a dome on top of a Stewart platform mounted on longitudinal and lateral rails on the ground, as shown in Figure 8.38(a) (Center for Computer-Aided Design, National Advanced Driving Simulator 1994). The motion system, on which the dome is mounted, provides horizontal and longitudinal travel and rotation in either direction, so the driver feels acceleration, braking, and steering cues as if he or she were actually driving a real car, truck, or bus. Inside the dome is a vehicle cab (or a full-sized vehicle body) equipped electronically and mechanically with instrumentation specific to its make and model. The 360-degree visual displays offer traffic, road, and weather conditions; a high-fidelity audio subsystem completes the driving experience. The driver is immersed in sight, sound, and movement so real that impending crash scenarios can be convincingly presented without the driver being endangered.

The key technology in a driving simulator is the real-time vehicle dynamic simulation (Bae and Haug 1987). The driver's interaction with the system through the steering wheel and gas and brake pedals is captured by sensors and electronics. The signals are converted into inputs to the underlying vehicle dynamic model. The equations of motion of the vehicle dynamic model must be solved faster than in real time, so that the actuators underneath the Stewart platform can perform the pushes or pulls that mimic various driving conditions. In the meantime, the simulation results provide large excursions in longitudinal and lateral directions that are used to give acceleration and motion cues to the driver inside. Without the real-time dynamic simulation, the driving experience cannot claim to be truly physics-based.

There are several notable driving simulators and associated research groups around the world, including those in the United Kingdom, France, Sweden, and the United States, in addition to the NADS in Iowa City (Figures 8.39(a) through (d)). Car manufacturers have built their own simulators in

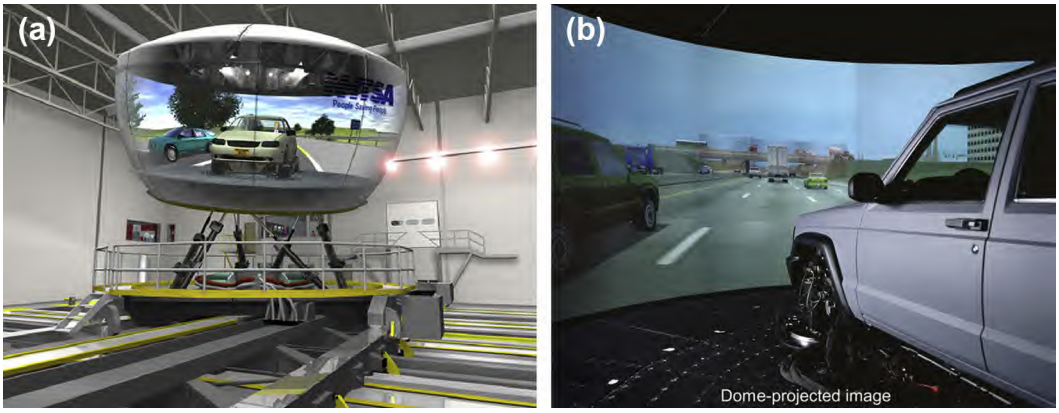


FIGURE 8.38

National Advanced Driving Simulator: (a) dome on top of motion platform mounted on rails and (b) vehicle cab and image projection inside the dome.

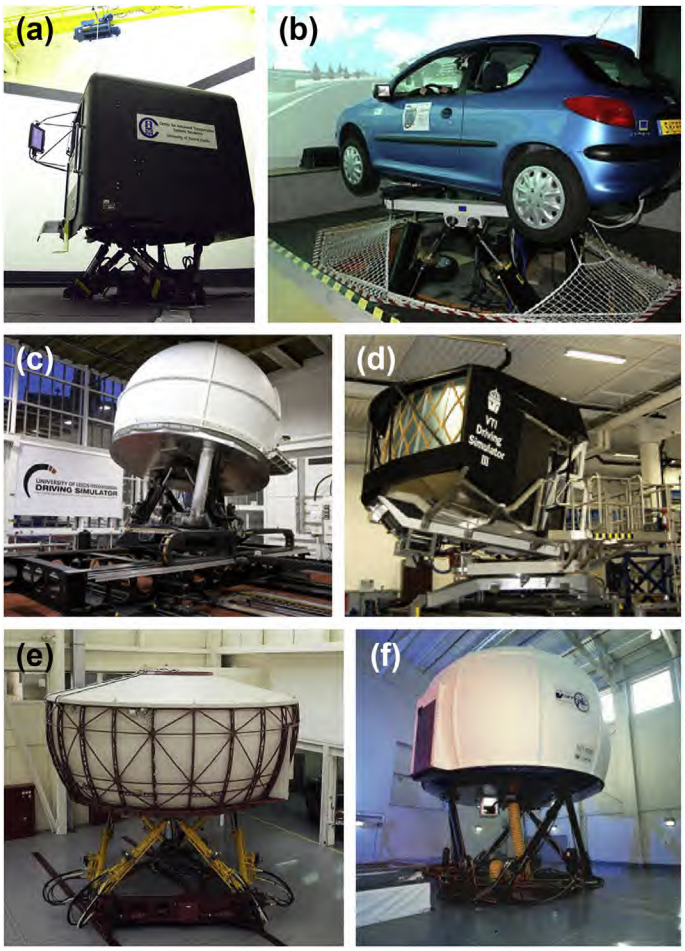


FIGURE 8.39

Driving simulators: (a) Center for Advanced Transportation Systems (University of Central Florida); (b) Valenciennes University (France); (c) Leeds University (United Kingdom); (d) Swedish National Road and Transport Research Institute; (e) Daimler-Benz (Berlin); (f) VIRTTEX (Ford Motor Co.).

the past 20 years, among them the pioneering Daimler-Benz Driving Simulator that was put into operation in 1985 and which has been enhanced since that time in many details and has been used in many different applications. Located in Berlin, this simulator went through a major modernization in 1994 (Figure 8.39(e)), including extension of the platform motion in a lateral direction to improve motion simulation quality. Ford Motor Company developed a motion-based driving simulator, called Virtual Test Track Experience (VIRTTEX) to test the reactions of sleepy drivers (Figure 8.39(f)).

The NADS in Iowa City was the largest and most advanced driving simulator in the world until December 2007, when Toyota announced that it had developed the current world's largest simulator at

its Higashifuji Technical Center in Susono City. Toyota's simulator houses an actual car on a platform inside a dome 15 feet tall and 23 feet wide. It includes a 360-deg. concave video screen that projects computer-generated images of roads, landscapes, street signs, and pedestrians. Also, it allows driving tests to be replicated under conditions that are too dangerous in the real world, such as the effects of drowsiness, fatigue, inebriation, illness, and inattentiveness.

8.6.4 RECREATIONAL WATERSLIDES

This case study involves verifying the safety of recreational waterslides using motion simulation. Safety is the top priority in the construction of recreational waterslides. Safety problems discovered after the slide is built and installed are usually too late and too costly to correct. In this study the riding object was assumed to be a particle with concentrated mass. Flume sections were represented in a CAD environment using geometric dimensions such as height and width. Friction forces between the riding object and the flume surface were also incorporated.

Basic sections of the flume, such as straight, elbow, and curved, serve as the building blocks for composing waterslide configurations (see Figure 8.40). In addition, guard sections (essentially vertical walls) are added to reinforce safety requirements, especially for elbow sections. The geometry of all sections is expressed in parametric surface forms in terms of the parametric coordinates u and w, using CAD geometric dimensions.

The overall waterslide configuration can be expressed mathematically as

$$\overline{\boldsymbol{X}}(u,w) = \sum_{i}^{N} \boldsymbol{X}^{i}\left(u^{i}, w^{i}\right) \tag{8.110}$$

where

$\boldsymbol{X}^{i}(u^{i}, w^{i})$ is the parametric equation of the ith flume section.
N is the total number of sections.

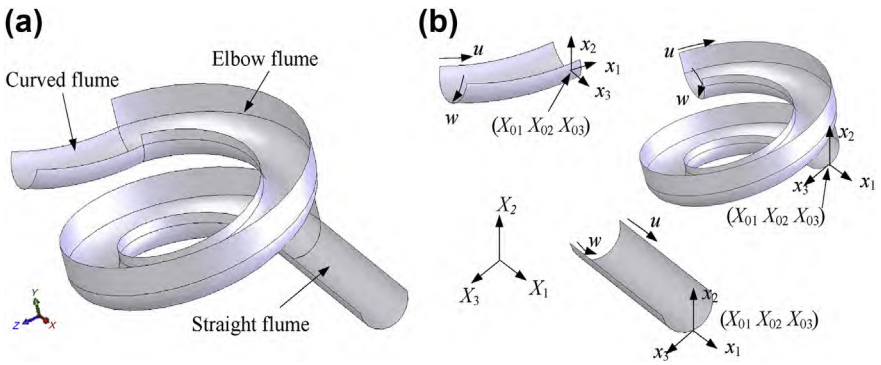


FIGURE 8.40

Geometric representation of a waterslide in flume sections: (a) assembled configuration and (b) individual flume sections.

For each flume section (with superscript i removed for simplicity),

$$\boldsymbol{X}(u,w) = [X_1(u,w), X_2(u,w), X_3(u,w)]^{\mathrm{T}} \tag{8.111}$$

where $X_j(u, w)$ is the jth coordinate of any given point on the surface with prescribed parameters (u, w). Note that these sections are translated and properly oriented to compose an overall waterslide configuration by the following translation and rotation operations:

$$\boldsymbol{X}(u,w) = \boldsymbol{X}_0 + \boldsymbol{T}(\theta)\boldsymbol{x}(u,w) = \begin{bmatrix} X_{0_1} \\ X_{0_2} \\ X_{0_3} \end{bmatrix} + \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} x_1(u,w) \\ x_2(u,w) \\ x_3(u,w) \end{bmatrix} \tag{8.112}$$

where

$\boldsymbol{T}(\theta)$ is the rotational matrix that orients the section by rotating through an angle θ about the X_2 (or Y) axis.
X_{0i} is the location of the local coordinate system of the section in the waterslide configuration.
$\boldsymbol{x}(u,w)$ is the surface function of the flume section referring to its local coordinate system.

As discussed in Section 8.2.1, the Lagrange equation of motion based on Hamilton's principle (Kane, 1985) for this particle dynamic problem, shown in Figure 8.41, can be stated as

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial L}{\partial \boldsymbol{q}} = \boldsymbol{Q} \tag{8.113}$$

where

The Lagrangian function L is defined as $L \equiv T - V$, $\dot{\boldsymbol{q}} = \partial \boldsymbol{q}/\partial t$.
The generalized coordinates $\boldsymbol{q}$ in this waterslide application are the parametric coordinates of the surface (i.e., $\boldsymbol{q} = [u, w]^{\mathrm{T}}$).

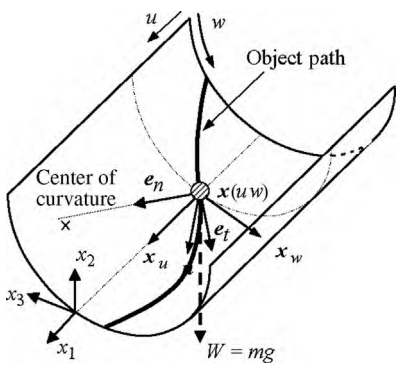


FIGURE 8.41

Object path and unit vectors for friction forces on a straight flume section.

When the system is conservative, $\boldsymbol{Q} = 0$. For a nonconservative system, $\boldsymbol{Q} = \boldsymbol{F}$, where $\boldsymbol{F}$ is the vector of generalized friction forces. For this motion analysis problem, the kinetic energy T and the potential energy U are, respectively,

$$T = \frac{m}{2}\left\|\dot{\boldsymbol{X}}(u, w)\right\|^2 \quad \text{and} \quad U = mgX_2(u, w) \tag{8.114}$$

where m is the particle mass and g is the gravitational acceleration.

For friction cases, the generalized friction forces $\boldsymbol{Q} = [f_u, f_w]^{\mathrm{T}}$ can be derived as (Chang, 2007)

$$f_u = -\mu(\boldsymbol{g} + \boldsymbol{a}_n)\cdot\boldsymbol{n}(\boldsymbol{e}_t \cdot X_{,u}), \quad \text{and} \quad f_w = -\mu(\boldsymbol{g} + \boldsymbol{a}_n)\cdot n(\boldsymbol{e}_t \cdot \boldsymbol{X}_{,w}) \tag{8.115}$$

where

μ is the friction coefficient.
$\boldsymbol{n}$ is the unit normal surface vector (shown in Figure 8.41).
$\boldsymbol{a}_n$ is the normal acceleration of the riding object.
$\boldsymbol{e}_t$ is the unit vector along the tangential direction of the object's path, which is also the direction of the object's velocity $\dot{\boldsymbol{X}}$ and the tangential acceleration $\boldsymbol{a}_t$.

Following Eq. 8.113, two coupled second-order ordinary differential equations that govern the particle motion can be obtained as

$$k_0\ddot{u} = k_1\dot{u}^2 + k_2\dot{w}^2 + k_3\dot{u}\dot{w} + k_4 \tag{8.116a}$$

$$k_0\ddot{w} = k_5\dot{u}^2 + k_6\dot{w}^2 + k_7\dot{u}\dot{w} + k_8 \tag{8.116b}$$

where k_0 through k_8 consist of polynomials of u and w and their products. Note that $\boldsymbol{X}$ must be at least second-order differentiable with respect to u and w. These requirements are satisfied within individual flume sections but not necessarily across sections.

The initial conditions, including initial position and velocity of the riding object, must be provided to solve the equations of motion:

$$u(0) = u^0, w(0) = w^0, \dot{u}(0) = \dot{u}^0, \quad \text{and} \quad \dot{w}(0) = \dot{w}^0 \tag{8.117}$$

The system of ordinary differential equations can be solved numerically for positions $u(t)$ and $w(t)$, velocities $\dot{u}(t)$ and $\dot{w}(t)$, and accelerations $\ddot{u}(t)$ and $\ddot{w}(t)$ of the riding object using, for example, Wolfram's *Mathematica* (1998).

A waterslide configuration consisting of 20 flume sections, shown in Figure 8.42, was modeled and analyzed (Chang 2008). The overall size of the waterslide was about 300 in. × 1150 in. × 378 in. Note that the riding object started at the center of the cross section ($w = 0.5$) of the top section. The friction coefficient was assumed to be $\mu = 0.08$.

The path of the riding object can be seen in Figure 8.42, which shows the object running over the edge of the flume section at three critical areas A, B, and C which posed a safety hazard to the rider. The design had to be revisited by either changing the composition of the configuration or using closed-flume (360 deg.) instead of open-flume sections (180 deg.) currently employed. The overall riding time was 21.2

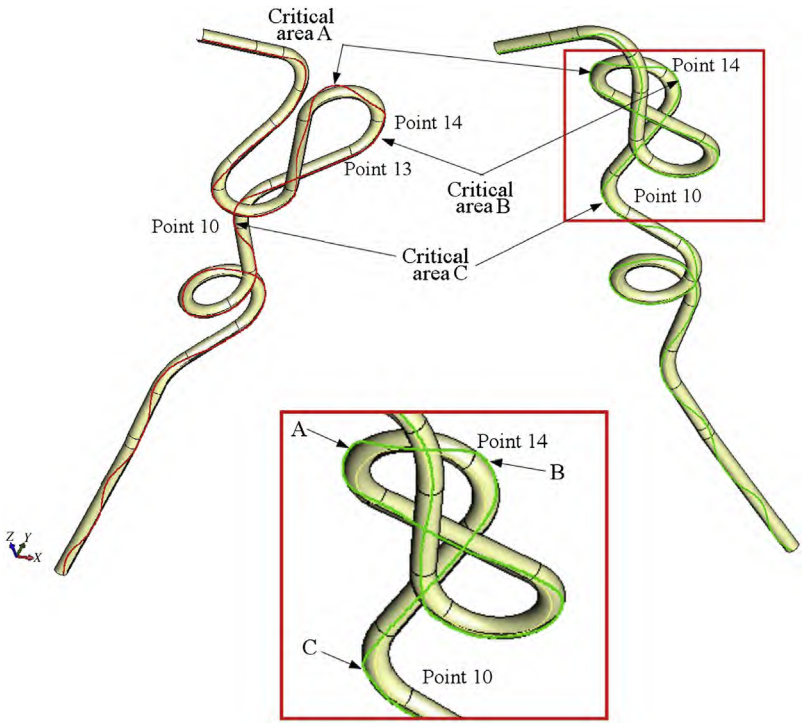


FIGURE 8.42

Object path on a large-scale waterslide showing critical areas of safety concern.

seconds which was very close to what was reported by the company that designed the waterslide (20 seconds). The maximum acceleration and velocity were 2.7 g and 12.8 mph, respectively; these are useful in characterizing the riding experiences (e.g., rider excitement level).

8.7 TUTORIAL EXAMPLES

A sliding block and a single-piston engine, mentioned in Section 8.1, are included in the tutorial lessons. The sliding block example is simply prepared for an easy start with both SolidWorks Motion and Pro/ENGINEER Mechanism Design. Default options and values are mostly used. Once readers are more familiar with either of these two software tools, they may move to the second tutorial example, the single-piston engine.

The first example simulates a block sliding down a 30-deg. slope with no friction. Because of gravity, the block slides and hits the ground, as depicted in Figure 8.43. Simulation results obtained from motion software can be verified using particle dynamics theory learned in a physics class.

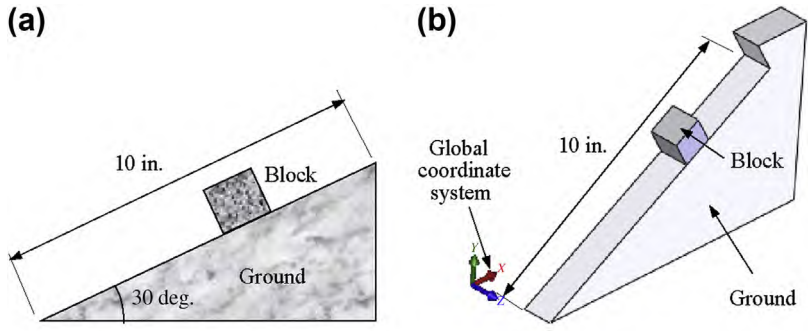


FIGURE 8.43

Sliding block: (a) schematic view and (b) motion model in CAD (SolidWorks).

8.7.1 SLIDING BLOCK

The physical model of a sliding block is very simple. The block was made of cast alloy steel with a size of 10 in. × 10 in. × 10 in. As shown in Figure 8.43, it traveled a total of 9 in. The units system employed for this example was IPS (inch, pound, second). The gravitational acceleration was 386 in./sec^2. The block was released from a rest position (that is, the initial velocity is zero).

The block and slope (or ground) were assumed rigid. A *limit distance mate* (in SolidWorks) was defined to prevent the block from sliding out of the slope face. The block reached the end of the slope face in about 0.3 sec, as indicated in Figure 8.44(a), which shows the *Y*-position of the mass center of

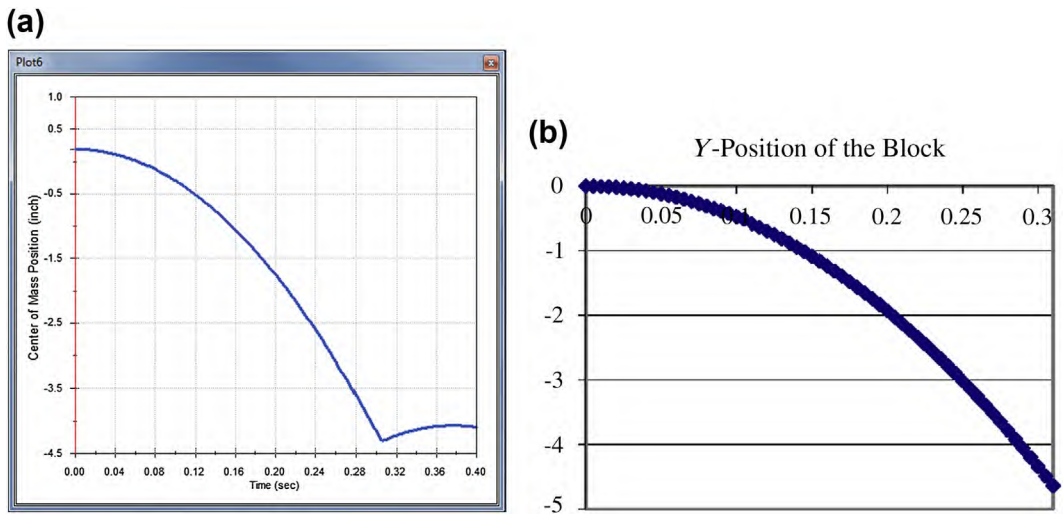


FIGURE 8.44

Graph of the *Y*-position of the mass center of the block, (a) obtained from SolidWorks Motion and (b) from spreadsheet calculations.

the block. Note that the graph shows that the block bounced back when it reached the end, which was because of the limit distance mate and was artificial. The *Y*-position of the mass center of the block was about 0.18 in. and traveled down to about −4.31 in. at 0.306 sec. The graph may be exported to an Excel file to check these numbers in SolidWorks Motion. The total vertical travel distance was 4.49 in.

The motion simulation result could be verified. Two assumptions had to be made to apply the particle dynamics theory to this sliding problem:

- The block was of a concentrated mass.
- No friction was present.

It is well known that the equations of motion for the sliding block can be derived from Newton's second law. By sketching a free-body diagram, we have the following acceleration equation:

$$F = ma = mg \sin 30 = 0.5mg; \text{hence } a = 0.5g \tag{8.118a}$$

The block's velocity and distance could be obtained by integrating Eq. 8.118a:

$$v = at = 0.5gt \tag{8.118b}$$

$$s = 0.5at^2 = 0.25gt^2 \tag{8.118c}$$

The *Y*-position of the block could be obtained as

$$P_y = -\frac{1}{2}at^2 \sin 30^\circ = -\frac{1}{8}gt^2 \tag{8.118d}$$

These equations could be implemented using, for example, a Microsoft® Excel spreadsheet for numerical solutions. The *Y*-position of the block, from 0 to 3.05 seconds is shown in Figure 8.44(b). This agreed very well with the SolidWorks Motion result in Figure 8.44(a). At time 3.05 sec, the *Y*-position of the block was −4.49 in., which matched well with that of SolidWorks Motion.

8.7.2 SINGLE-PISTON ENGINE

The second example is the kinematic analysis of a single-piston engine. The engine consisted of four major components: case, propeller, connecting rod, and piston, as shown in Figure 8.45(a). The propeller was driven by a rotary motor at the angular speed of 60 rpm (i.e., one revolution per second). No gravity was present and the English units system was assumed. The engine was properly assembled with one free degree of freedom. When the propeller was driven by the rotary motor, it rotated, the crank shaft drove the connecting rod, and the connecting rod pushed the piston up and down within the piston sleeve.

The engine assembly consisted of three subassemblies (engine case, propeller, and connecting rod) and one part (piston). The engine case was fixed (ground body). The propeller was assembled to the engine case using concentric and coincident mates, as shown in Figure 8.45(b). It was free to rotate along the *X*-direction. The connecting rod was assembled to the propeller (at the crankshaft) using concentric and coincident mates. It was free to rotate relative to the propeller (at the crankshaft) along the *X*-direction. Finally, the piston was assembled to the connecting rod (through the piston pin) using a concentric mate. The piston was also assembled to the engine case using another concentric mate. This mate restricted the piston's movement along the *Y*-direction, which in turn restricted the top end of the connecting rod to move vertically.

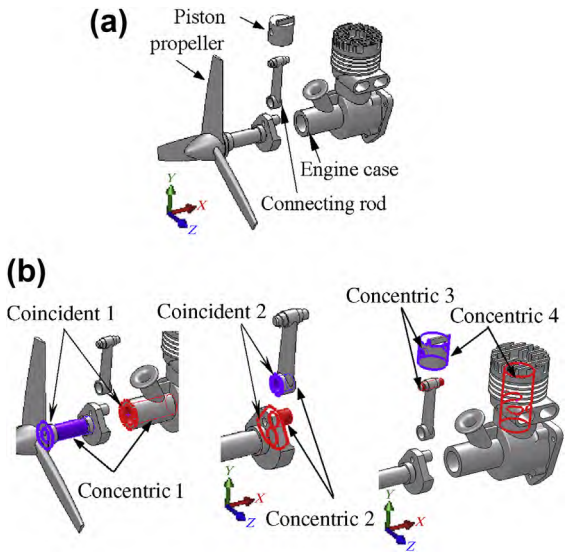


FIGURE 8.45

Single-piston engine: (a) exploded view and (b) constraints defined between bodies (or subassemblies).

The position and velocity of the piston obtained from motion analysis are shown in Figures 8.46(a) and (b), respectively. From the graph, we see that the piston moved between about 1.0 and 2.1 in. vertically. The total travel distance was about 1.1 in., which could be easily verified by the radius of the crankshaft, which was 0.58 in. The piston travel distance was two times the radius of the crankshaft, which was 1.16 in.

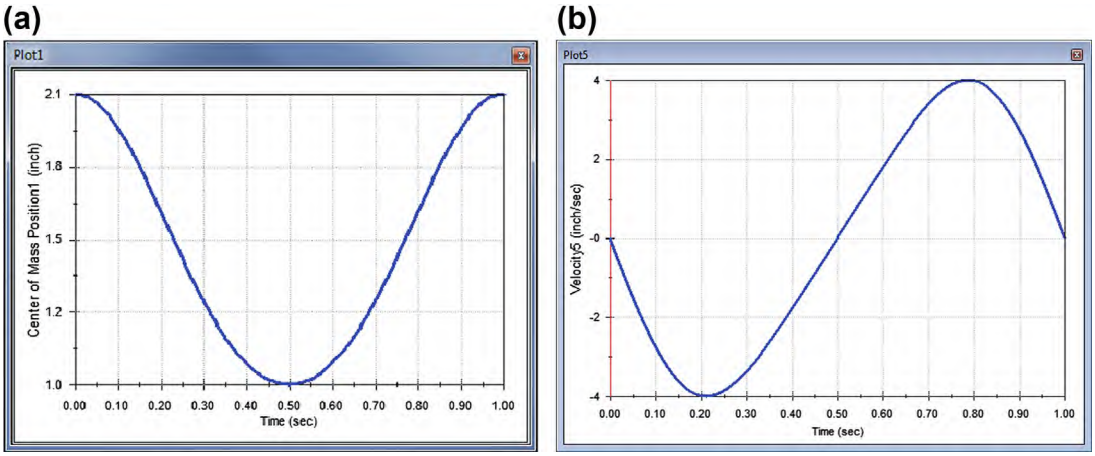


FIGURE 8.46

Result graphs: (a) *Y*-position of the piston and (b) *Y*-velocity of the piston.

8.8 SUMMARY

In this chapter, we discussed methods for motion analysis, both analytical and computer-aided. Analytical methods employ numerous concepts and physics laws to support dynamic analysis for relatively simple problems. They are not adequate to support general problems encountered in engineering design. Even for a two-rigid-body problem, as discussed in Example 8.5, in which it is extremely difficult, if not entirely impossible, to formulate the equations of motion. However, analytical methods are important for mechanical engineers to understand the fundamentals of motion analysis and to be able to develop references for verifying motion analysis results obtained from computer software.

On the other hand, computer-aided methods are formulated in a general and systematic setting. Any mechanical systems that are characterized by bodies and kinematic joints can be formulated in the same type of differential (and algebraic) equations. No matter how complex the motion problem, it can be solved numerically using computers.

A number of software tools are available for support of motion simulation, including codes imbedded in CAD, standalone codes with CAD connections, and specialized codes. It is hoped that this chapter has provided the reader with a good understanding of how the motion analysis method works, how to create motion analysis models, and how to choose the right software tool for the problem at hand.

The chapter also presented case studies involving design aspects of motion analysis, including a Formula SAE racecar and an HMMWV suspension. These two cases should provide a general idea of the applications that lend themselves to simulation for motion analysis and design in general.

QUESTIONS AND EXERCISES

8.1. Derive equations of motion for the particle sliding along a straight line shown in Figure 8.3(b) using Newton's method. Compare your results with those obtained in Example 8.2.

8.2. Repeat Exercise 8.1; that is, derive equations of motion for the particle shown in Figure 8.3(b) using Newton's method as well as Lagrange's equations, assuming that friction coefficient μ is nonzero.

8.3. Equation 8.10 was derived with an assumption of no friction. If friction is present, derive the generalized forces $\boldsymbol{Q}$ for the particle sliding along the curve $\boldsymbol{x}(u)$.

8.4. Repeat Example 8.3 using an initial angular velocity that is greater than $\omega_{\min} = \sqrt{\frac{g}{r_0}}$, for example, $\omega_0 = \sqrt{\frac{8g}{3r_0}}$.

8.5. Derive the acceleration equations for the slider-crank mechanism by taking derivatives of Eqs 8.50a and 8.50b with respect to time. Solve these equations for the linear acceleration of the piston and the angular acceleration of the mate Concentric 2, using a spreadsheet.

8.6. Show that $\boldsymbol{\Phi}_{\mathbf{q}}$ of Eq. 8.81 in Example 8.8 is nonsingular.

8.7. Solve for $\ddot{\boldsymbol{q}}$ of Eq. 8.83.

8.8. Several driving simulators were mentioned in Section 8.6.3. Please review three or more, identify their main uses, and compare their strengths and weaknesses.

REFERENCES

Bae, D.S., Haug, E.J., 1987. A recursive formulation for constrained mechanical systems. Part I: Open loop. Mechanics Based Design of Structures and Machines. An International Journal 15 (3), 359–382.

Chang, K.H., 2007. Computer-Aided Modeling and Simulation for Recreational Waterslides. Mechanics Based Design of Structures and Machines. An International Journal 35 (3), 229–243.

Chang, K.H., 2008. Modeling and Simulation for Waterslides in CAD Flume Sections. International Journal of Pure and Applied Mathematics 42 (3), 345–352.

Chang, K.H., Joo, S.-H., 2006. Design Parameterization and Tool Integration for CAD-Based Mechanism Optimization. Advances in Engineering Software 37, 779–796.

Greenwood, D.T., 1987. Principles of Dynamics. Prentice Hall.

Haug, E.J., 1989. Computer Aided Kinematics and Dynamics of Mechanical Systems, vol. 1. Basic Methods. Allyn & Bacon.

Kane, T.R., Levinson, D.A., 1985. Dynamics: Theory and Applications. McGraw-Hill.

NADS Vehicle Dynamics Simulation, Release 4.0, 1994. Center for Computer-Aided Design. In: National Advanced Driving Simulator. University of Iowa.

Stewart, D.A., 1965. Platform with six degrees of freedom. Proceedings of the Institute of Mechanical Engineering 180 (15, part I), 371–387.

U.S. Tank-Automotive Research and Development Command, 1979. NATO Reference Mobility Model, Edition I. Technical Report 12503 prepared for the North Atlantic Treaty Organization.

Wheeler R.M. III. 2006. Vehicle Dynamic Simulation and Validation of a Formula SAE Car. In: M.S. Thesis. The University of Oklahoma: Norman, OK.

Wolfram, S., 2003. The Mathematica Book, fifth ed., ISBN 1-57955-022-3 (reference.wolfram.com/legacy/v5_2/).

CHAPTER

FATIGUE AND FRACTURE ANALYSIS

9

CHAPTER OUTLINE

9.1 Introduction 464
9.2 The Physics of Fatigue 467
9.3 The Stress-Life Approach 470
9.3.1 The S-N Diagram 470
9.3.2 Nonfully Reversed Cyclic Loads 472
9.3.3 In-Phase Bending and Torsion 475
9.3.4 Complex Multiaxial Stress 476
9.3.5 Cumulative Damage 477
9.4 The Strain-Based Approach 478
9.4.1 The Manson–Coffin Equation 478
9.4.2 Multiaxial Analysis 483
9.5 Fracture Mechanics* 484
9.5.1 Basic Approaches 485
9.5.2 Linear Elastic Fracture Mechanics 486
9.5.3 Mixed Mode 488
9.5.4 Quasistatic Crack Growth 491
9.5.5 The Extended Finite Element Method 492
9.6 Dynamic Stress Calculation and Cumulative Damage 497
9.6.1 Dynamic Stress Calculations 497
9.6.2 Peak-Valley Editing 500
9.6.3 Rain-Flow Counting 501
9.6.4 Blocks to Failure 504
9.7 Fatigue and Fracture Simulation Software 505
9.7.1 General-Purpose Codes for Crack Initiation 505
9.7.2 Non-FEA-Based Crack Propagation 506
9.7.3 FEA-Based Crack Propagation 507
9.8 Case Studies and Tutorial Example 508
9.8.1 Case Study: Tracked Vehicle Roadarm 508
9.8.2 Case Study: Engine Connecting Rod 511
9.8.3 Tutorial Example: Crankshaft 516
9.9 Summary 518
Questions and Exercises 518
References 519

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00009-0
Copyright © 2015 Elsevier Inc. All rights reserved.

One of the most technically challenging issues facing aerospace and mechanical engineers is structural failure due to fatigue and fracture, which causes mechanical failures and safety hazards. Because of the lack of adequate simulation tools for crack growth analysis, especially for complex three-dimensional structural components, heavy emphasis has been placed on physical testing, which is costly and time consuming. Very recently, computational methods were expanded to support fatigue and fracture simulations, especially the finite element method (FEM) and other advanced methods, such as extended FEM (XFEM). It is important to understand and incorporate crack initiation and crack growth simulation techniques into the design of structural components.

In general, fatigue and fracture occur in three stages: crack initiation, crack propagation, and fracture. As design engineers, the common questions we should ask are:

- When and where cracks initiate and after how many load cycles.
- How a crack grows—in which direction and at what rate.
- When the crack size reaches a critical point where the structure is too much damaged to withstand the operating load, leading to fracture.

Different methods have been developed to predict fatigue and fracture for all three stages, with the aim of answering the questions stated above. Some are empirical and largely rely on experimental results in the form of graphs and data. These methods are usually simple and can be carried out by hand or by using lookup tables when nominal stress is calculated. However, they are usually limited to simple cases in terms of loads, structural geometry, and crack size and type. More theoretically sound approaches involve more computations, usually providing more accurate predictions. They also are able to support more general applications. It is important to note that there is no single theory or method that is universally superior to the rest. All involve certain limitations under prescribed assumptions.

This chapter provides a brief overview on the computational methods that help us answer questions about structural fatigue and fracture in various stages. The focus here is not a thorough review of fatigue theory but a discussion of the essential elements of fatigue and fracture simulations from a practical perspective. The goal of this chapter is to provide confidence and competency in the use of software tools for creating adequate models and obtaining reasonable results to support design. Readers who are interested in fatigue and fracture theory can refer to excellent treatments such as Collins (1993). Simple examples are introduced to illustrate the theory and computational methods. In addition, practical examples are introduced, including a tracked vehicle roadarm and an engine connecting rod, to illustrate and demonstrate computational methods for structural fatigue and fracture analyses. Overall the objectives of this chapter are as follows to provide basic fatigue and fracture theory using simple examples to promote understanding of how the theory works; to promote familiarity with fatigue and fracture modeling and computations for effective use of existing commercial software tools; to promote use of either Pro/MECHANICA® Structure or SolidWorks® Simulation for basic applications (after the tutorial lessons have been completed).

9.1 INTRODUCTION

Failure of structural components due to fatigue and fracture is a major issue that spans several engineering disciplines and costs hundreds of billions of dollars (National Bureau of Standards, 1983). Structural components commonly observed in the aerospace and mechanical engineering fields are obvious examples where crack growth can lead to downtime or failure and may even result in substantial damage and loss of life.

It is widely recognized that about 80% of the failures of mechanical/structural components and systems are related to fatigue (Bannantine et al., 1990). Structural fatigue has produced many losses of aging aircraft. One of the most well known is Aloha Airlines Flight 243, a scheduled flight between Hilo and Honolulu in Hawaii on a Boeing 737-297. On April 28, 1988 the aircraft suffered extensive damage after an explosive decompression in flight (National Transportation Safety Board, 1989). Nearly 6 meters of cabin skin and structure aft of the cabin entrance door and above the passenger floor line separated from the aircraft (Figure 9.1(a)) while cruising at 24,000 feet. One flight attendant was sucked from the airplane and another 65 passengers and crew members were injured. Amazingly, the plane made a safe emergency landing at Kahului Airport on Maui. The subsequent investigation found that debonding and fatigue damage had led to the failure. The aircraft involved had completed 89,680 flight cycles with an average flight time of only 25 minutes, almost all of them in the marine environment of the Hawaiian Islands—a somewhat atypical service life that was considered to have allowed corrosion to increase the likelihood of fatigue.

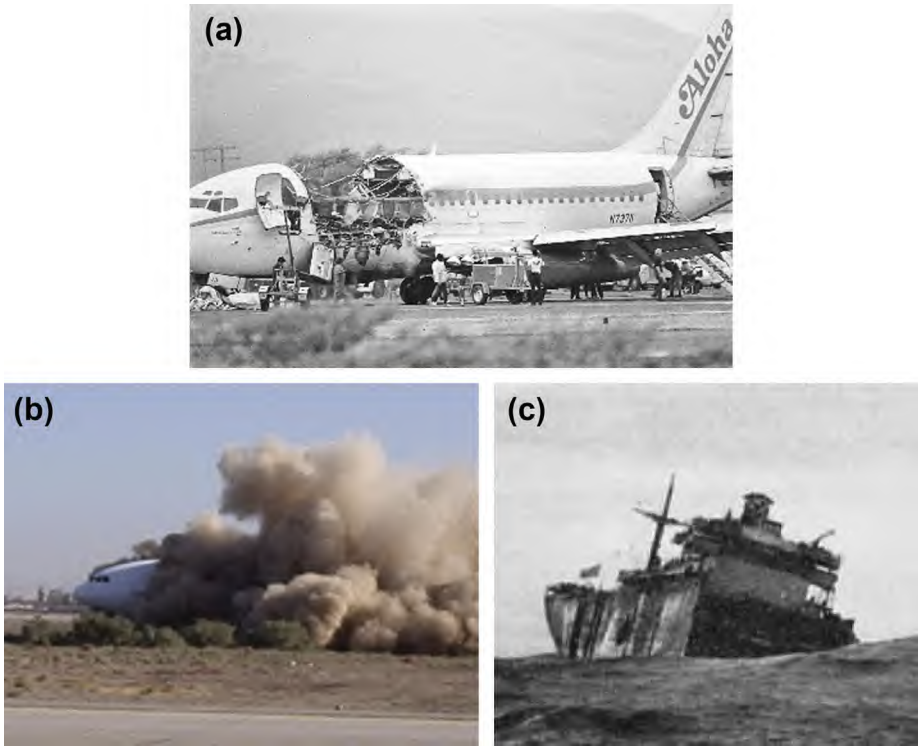


FIGURE 9.1

Major structural failures due to fatigue and fracture: (a) Aloha Airlines Flight 243 lost a third of its roof due to a stress fracture (source: Chapter 7; Bernard J. Hamrock, Steven R. Schmid, Bo O. Jacobson, Fundamentals of Machine Elements, 2nd Ed., McGraw-Hill, 2005); (b) United Airlines Flight 232 broke up during an emergency landing on the runway at Sioux City, Iowa; (c) The *John P. Gaines* split in two off the Aleutians in 1943 (source: http://www.ntsb.gov/policy/policies.html).

Another incident involved United Airlines Flight 232, scheduled from Stapleton International Airport in Denver to O'Hare International Airport in Chicago. On July 19, 1989, the Douglas DC-10 suffered an uncontained failure of its number 2 engine, resulting in the crew's inability to move the flight controls (National Transportation Safety Board, 1990). Only the thrust levers for the two remaining engines worked. The aircraft eventually broke up during an emergency landing on the runway at Sioux City, Iowa (Figure 9.1(b)), killing 111 of its 285 passengers and one of 11 crew members. The investigation attributed the cause of the accident to a failure of United Airlines maintenance processes to detect an existing fatigue crack at an inspection prior to flight.

Fatigue and fracture cause problems not only in aircrafts. Liberty ships (Bannerman and Young, 1946) were lightly armed cargo vessels built in the United States for transporting desperately needed supplies across the U-boat-infested Atlantic to a beleaguered Europe during World War II. Some 2,700 vessels were built from 1942 until the end of the war. They suffered hull and deck cracks, and a few were lost to such structural defects. During the war, there were nearly 1,500 instances of significant brittle fractures. Twelve ships broke in half without warning, including the *John P. Gaines* (Figure 9.1(c)), which sank on November 24, 1943, with the loss of ten lives.

Fatigue and cracks are critical issues to be addressed in product design, especially for products that are to endure heavy operating loads in a highly repetitive manner. Even though it is impractical to design a structure that is fatigue and fracture free, the design should be durable to prolong the service life of the structural components.

The two terms—*fatigue* and *fracture*—have been used interchangeably. In general, fatigue and fracture occur in three stages: crack initiation, crack propagation, and fracture failure. In this chapter, we specifically refer to *fatigue* at the crack initiation stage and keep *fracture* for crack propagation and fracture failure. Fatigue begins with microcracks usually due primarily to local stress concentration and the pile-up of dislocations. Local stress concentrations occur around pores, inclusions, or impurities, unsmooth surfaces, and the like. Cracks can also be caused by a local decrease in fatigue strength due to a pile-up of dislocations, which form slip bands that grow and lead to cracks. Which of these two mechanisms dominates depends on the purity of the material, the nature of the loading, and so forth.

Crack initiation refers crack growth up to 0.08 in. (2 mm). Usually such small cracks do not affect the functionality and performance of structural components. However, cracks propagate after they are initiated since the geometry of a crack produces a very high concentration of stress at the crack tip. The crack continues to grow in a stable and predictable manner, which is referred to as crack propagation. When the crack reaches a critical size that is over the material strength, in this case the fracture toughness, it grows very fast and unstably, leading to sudden and unexpected catastrophic fracture failures. Note that for brittle materials there is no clear crack initiation. Once the crack initiates, it grows steadily; that is, it directly enters the crack propagation stage.

The questions are, first, when and where in the structure a crack initiates, and how many load cycles it takes for the component to reveal initial cracks; second, how the crack propagates, in which direction, and at what rate. The answers are critical since they offer the basis for setting up inspection and maintenance schedules. The third question is when does the crack grow to a point where the structure is no longer able to withstand the loads, leading to fracture that causes catastrophic failure.

We discuss theory and computation methods that help us answer these critical questions for designing durable structural components. The discussion is brief, just enough to provide the basics of effective software tool use. Software tools that employ some of this theory and associated methods are introduced. In most of the discussion, we assume that finite element analysis (FEA) is employed for

stress and strain calculation, where geometric stress concentration factors have been incorporated. We review first stress-based methods for high-cycle fatigue and then strain-based methods for low-cycle fatigue. Both predict fatigue life of the structure up to crack initiation. Theory and methods for crack propagation, such as linear elastic fracture mechanics (LEFM), which provides crack propagation rate and direction, are discussed. We also discuss commercial software tools. Finally, example problems modeled and solved using some of the commercial codes are presented.

9.2 THE PHYSICS OF FATIGUE

In materials science, fatigue is the progressive and localized structural damage that occurs when a material is subjected to cyclic loading. Usually, the maximum stress values are much less than the ultimate tensile strength and much below the yield strength of the material. This phenomenon was recognized in the mid-1800s; that is, that metal under a repetitive or fluctuating load fails at a stress level lower than required to cause failure under a single application of a similar load.

Why does metal fatigue under such low stress? When a component such as that shown in Figure 9.2 is subjected to a uniform sinusoidally varying force for a period of time, a crack can be seen to initiate on the circumference of the component. This crack propagates through the component, as illustrated in the figure, until the remaining intact section is incapable of sustaining the imposed stresses, in particular at the crack tip. At that point the component fails.

The physical development of a crack is generally divided into three stages: crack initiation (stage 1), crack growth or propagation (stage 2), and fracture (stage 3). Fatigue cracks initiate through the release of shear strain energy. Shear stresses result in local plastic deformation along slip planes or slip bands, as shown in Figure 9.3. As the loading is cycled sinusoidally, the slip planes move back and forth like a pack of cards, resulting in small extrusions and intrusions on the crystal surface (Halfpenny, 2005). These surface disturbances are approximately 1 to 10 microns in height and constitute embryonic cracks.

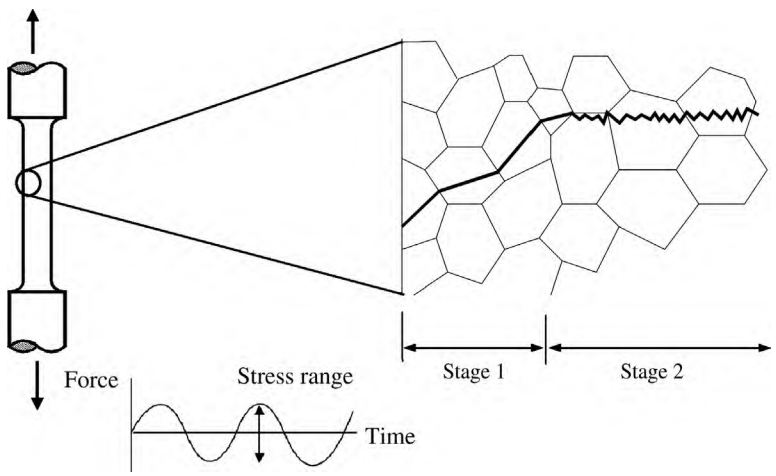


FIGURE 9.2

Component under a uniform sinusoidally varying force.

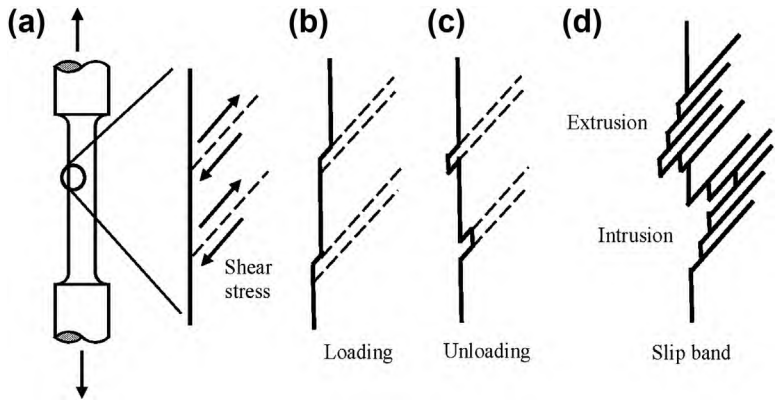


FIGURE 9.3

Physics of crack initiation: (a) component under a cyclic load, (b) slip under loading, (c) unloading, and (d) slip band form along planes of maximum shear giving rise to surface extrusion and intrusions.

A crack initiates in this way until it reaches the grain boundary. At this point the mechanism is gradually transferred to the adjacent grain. When the crack has grown through approximately 3 grains, it changes its direction of propagation. Stage 1 growth follows the direction of the maximum shear plane, or 45 deg. to the direction of loading. During stage 2 the physical mechanism of fatigue changes. The crack is now sufficiently large to form a geometrical stress concentration. At this point, a tensile plastic zone is created at the crack tip and the crack propagates perpendicular to the direction of the applied load. When a crack grows to a critical size, at which the structure is not able to withstand the next cyclic load, fracture (stage 3) occurs.

Just as the physical mechanism of fatigue is divided into stages, the methods of analysis are conventionally divided into stages. Stage 1 is typically analyzed using the stress-strain (*S-N*) or local strain (ε-*N*) approach; stages 2 and 3 are analyzed using a fracture mechanics-based approach.

Even though *S-N* analysis is widely used in test-based fatigue analysis, it has one major drawback for fatigue life computation. Fatigue initiation is driven by local plastic strains, but *S-N* analysis uses elastic stress as the input; therefore, it is often limited to components with minimum and yet limited local plasticity areas. Such components often reveal high-cycle fatigue; that is, the load cycle is greater than 10^3 before cracks are initiated. On the other hand, when local strains are significant, the strain-life (or ε-*N*) approach is more suitable. In these cases, fatigue life is often less than 10^3, which is so-called low-cycle fatigue. Both approaches are discussed in this chapter (Sections 9.3 and 9.4, respectively).

A complete fatigue prediction analysis can use a combination of both methods:

$$N = N_i + N_g \tag{9.1}$$

where

N is total fatigue life.
N_i is fatigue life to initiation.
N_g is life taken to propagate the crack to failure.

However, most engineering components are at either one stage or the other. In this case, it is normal to conservatively consider only one stage. For example, in most ground vehicle or heavy equipment designs, the life of suspension components is typically governed by time to crack initiation. The components are relatively stiff and the load that the components bear is relatively heavy. Once the crack has initiated, it takes a relatively short time to propagate to failure.

By contrast, many aerospace applications such as skin panels of cowling of aircrafts use flexible components made of ductile materials. In this case, cracks propagate relatively slowly and so fracture mechanics approaches are usually more appropriate. Fatigue and fracture calculations normally assume that the structural component is under cyclic loading—that is, repetitive loads with the same magnitudes—as shown in Figure 9.4.

The most basic cyclic load is fully reversed (Figure 9.4(a)), which is often seen in fatigue experiments in a laboratory setup, where the material specimen is loaded with pure bending moment that reveals the identical stress magnitudes in tension (σ_{max}) and compression (σ_{min}). The mean (average) σ_m stress is therefore 0 and the alternating stress is $\sigma_a = 1/2(\sigma_{max} - \sigma_{min}) = \sigma_{max}$. This fully reversed cyclic load is also seen in machinery such as shafts rotating at constant speed with load exerted by mechanical components—a paired gear or driving belt, for example.

The second kind of cyclic load is nonfully reversed, where the magnitude of the maximum and minimum stresses are different. This can be like the repeated load shown in Figure 9.4(b), where the load is applied and released in one cycle ($\sigma_{min} = 0$), or it can be similar to a fully reversed load but with nonzero mean stress, as shown in Figure 9.4(c). The most general and complicated load is random, as shown in Figure 9.4(d). This kind of load appears in many applications; e.g., suspension components of a ground vehicle that is driven over a bumpy road.

It is important to understand the effect of loads on fatigue life calculations. First, load magnitude affects stress magnitudes. Any inaccuracy in stress values amplifies inaccuracy in fatigue life prediction. Second, most of the experimental data obtained from fatigue tests are obtained under uniaxial loads, which lead to uniaxial stress in the structure, such as uniaxial normal stress or shear stress. In practice, structural components are usually under multiaxial loads, yielding a combination of normal and shear stresses. Blindly applying theory developed for uniaxial stress cases to multiaxis stresses can result in erroneous fatigue prediction that contributes to incorrect design decisions.

In Sections 9.3 through 9.5 we assume cyclic loads while discussing basic fatigue and fracture theories. Methods discussed in Section 9.6 assume general random loads.

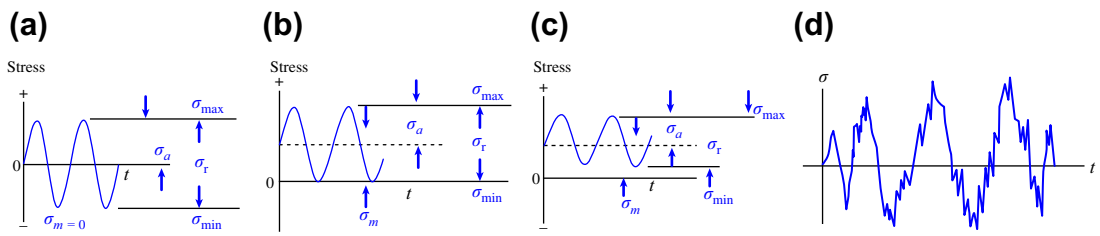


FIGURE 9.4

Cyclic loads: (a) fully reversed; (b) nonfully reversed, repeated; (c) nonfully reversed, fluctuated; (d) random load.

9.3 THE STRESS-LIFE APPROACH

For crack initiation calculations, there are stress-based and strain-based approaches. Stress-based methods are in general applicable to high-cycle fatigue, where fatigue life is usually between 10^3 and 10^6, which is the endurance limit of the material. The endurance limit is a material property, beyond which the structure is deemed to be of little or no fatigue concern. Theoretically, fatigue life is infinite for such structures. The endurance limit varies for different materials, although 10^6 is the most common case and is often assumed if no better data are at hand. This limit varies according to factors (Juvinall and Marshek, 2005) including surface finish, cross-sectional shape of the structural component, type of load, and so forth. This limit has to be corrected from published data, obtained from laboratory tests under a controlled environment, to a lower value by incorporating these factors.

In this section we discuss the *S-N* diagram for fatigue life prediction of high-cycle fatigue cases. The basic *S-N* diagram assumes uniaxial stress under fully reversed cyclic load, which is relatively limited in applications. Methods that extend the *S-N* for applications other than fully reversed cyclic loads are also included in this subsection. Note that stress-life analysis is the simplest and most widely used approach, providing a quick and rough estimate of the fatigue life of structural components.

9.3.1 THE S-N DIAGRAM

The *S-N* diagram is the most basic method for fatigue life prediction of high-cycle fatigue, where fatigue life N is $10^3 < N < 10^6$. It is essentially based on experimental data, mostly under uniaxial, fully reversed cyclic loading. The loading can be bending or torsion. Most of the data and graphs available in textbooks and handbooks are for metals, which are assumed to be homogeneous and isotropic. Some are available for composite materials; however, in addition to fatigue crack, delamination between material layers is a critical issue to be addressed. We assume homogeneous and isotropic materials in this section.

A typical *S-N* diagram, shown in Figure 9.5, is on a log-log scale. The vertical axis is the fatigue strength S_f of the material, and the horizontal axis is the fatigue life N. There are two important fatigue strengths on the vertical axis: S_{10^3} and S_e. S_e is the endurance limit of the material, which is probably the most important material property in high-cycle fatigue calculations because, theoretically (this is backed up by experimental data), if the maximum operating stress of the structural component is less than the endurance limit, the component is said to have infinite fatigue life. That is, fatigue life N is greater than 10^6.

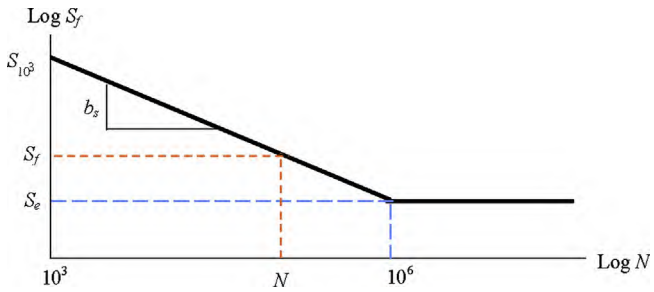


FIGURE 9.5

Standard S-N diagram for high-cycle fatigue.

As designers, we prefer a stress level lower than the endurance limit whenever possible. In general, for steels under pure bending, $S_e = 0.5S_{ut}$, where S_{ut} is the ultimate tensile strength. The endurance limit can be different for stress under loads other than pure bending. For steels under axial or torsion stress, the endurance limit is $S_e = 0.45S_{ut}$ and $S_e = 0.29S_{ut}$, respectively. This relation is true for a controlled environment in laboratory tests. It must be corrected by incorporating factors such as size and load, to support the specific operating scenarios for the subject applications. The reader should refer to Juvinall and Marshek (2005) for more details. We assume that the notation S_e represents the corrected endurance limit.

The other property, S_{10^3}, is the fatigue strength corresponding to $N = 10^3$. This is the lower limit of the valid range of high-cycle fatigue. In general, for steels under bending, $S_{10^3} = 0.9S_{ut}$. Under axial and torsion stress, the fatigue strength is $S_{10^3} = 0.75\mathrm{S}_{ut}$ and $S_{10^3} = 0.72S_{ut}$, respectively. If the maximum stress of a structural component is between these two strength properties (i.e., $S_{10^3} < \sigma_{\max} < S_e$), the fatigue life of the component is finite—$10^3 < N < 10^6$. In this case, the fatigue life corresponding to the maximum stress $\sigma_{\max}$ can be obtained by

$$N = \left(10^{-\overline{C}}\sigma_{\max}\right)^{\frac{1}{b_s}} \tag{9.2a}$$

where

b_s is the slope of the fatigue line (straight line) in the *S-N* diagram shown in Figure 9.5, called the Basquin exponent.
$\overline{C}$ is the intersect of the fatigue line with the vertical axis.

Both b_s and $\overline{C}$ can be obtained, respectively, by

$$\overline{C} = \log\frac{(S_{10^3})^2}{S_e} \tag{9.2b}$$

and

$$b_s = -\frac{1}{3}\log\frac{S_{10^3}}{S_e} \tag{9.2c}$$

One common mistake made in using these equations, especially Eq. 9.2b, is mixed stress and strength units. If MPa is used for stress and strength, its use must be consistent. If MPa is mixed with Pa (e.g., MPa for S_{10^3} and Pa for S_e), $\overline{C}$ and b_s will be incorrect, which affects the fatigue life calculation in Eq. 9.2a.

Equation 9.2a allows prediction of fatigue life of the structural component when the maximum stress has been calculated. This is often referred to as an analysis problem. On some occasions, we may want to find the allowable stress for a required fatigue life. The allowable stress σ_{all} can be obtained by

$$\sigma_{\text{all}} = \frac{S_f}{n} \tag{9.3a}$$

where n is the safety factor and

$$S_f = 10^{\overline{C}}N^{b_s} \tag{9.3b}$$

and N is the desired or required fatigue life. This is in general a design problem.

The value of b_s, for a particular *S-N* slope, provides a good indication of how accurate the estimate of stress needs to be to give a reliable life prediction. If b_s is 10 then a 7% error in stress causes a 100% error in fatigue life. Even if the stress value is calculated accurately using FEA, if the load magnitude is slightly inaccurate, the stress is affected proportionally (assuming linear elastic), which impacts the life estimate exponentially. Therefore, load must be accurately captured to carry out a reliable fatigue life analysis.

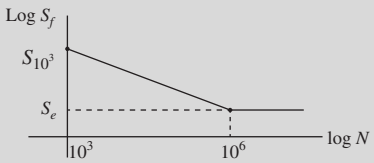
EXAMPLE 9.1

A steel has a tensile strength of $S_{ut} = 90{,}000$ psi and a corrected endurance limit of $S_e = 33{,}000$ psi for a machined surface. Assume that the fatigue strength at $N = 10^3$ is $S_{10^3} = 0.9S_{ut}$.

(a) Draw an *S-N* diagram with all needed data marked.
(b) Calculate coefficients $\overline{C}$ and b_s for the equation of the finite life and fatigue strength (i.e., Eq. 9.3b).
(c) Calculate the fatigue strength for $N = 10{,}000$.
(d) If the maximum stress is 34.2 ksi, calculate the corresponding fatigue life *N*.

Solution

(a) The *S-N* diagram can be drawn as shown below, with fatique strength for $N = 1{,}000$ (i.e., S_{10^3}) and endurance limit S_e calculated below.



$$S_{10^3} = 0.9\, S_{ut}$$
$$= 0.9(90{,}000) = 81{,}000 \text{ psi} = 81 \text{ ksi}$$
$$S_e = 33{,}000 \text{ psi} = 33 \text{ ksi}$$

(b) b_s and $\overline{C}$ can be calculated using Eq. 9.2b and 9.2c, respectively, as follows

$$b_s = -\frac{1}{3}\log\frac{S_{10^3}}{S_e} = -\frac{1}{3}\log\frac{81}{33} = -0.1300$$

$$\overline{C} = \log\frac{[S_{10^3}]^2}{S_e} = \log\frac{(81)^2}{33} = 2.298 \text{ (in ksi)}$$

(c) The fatigue strength for $N = 10{,}000$ can be calculated using Eq. 9.3b as

$$S_f = 10^{\overline{c}}(N)^{b_s} = 10^{2.298}(10{,}000)^{-0.1300} = 59.98 \text{ ksi}$$

(d) The fatigue life for a given stress can be obtained using Eq. 9.2a as

$$N = \left(\frac{S_f}{10^{\overline{c}}}\right)^{\frac{1}{b_s}} = \left(\frac{\sigma}{10^{\overline{c}}}\right)^{\frac{1}{b_s}} = \left(\frac{34.2}{10^{2.298}}\right)^{\frac{1}{-0.1300}} = 7.29 \times 10^5 \text{ cycles}$$

9.3.2 NONFULLY REVERSED CYCLIC LOADS

S-N diagrams assume fully reversed cyclic load or stress. For cases of nonfully reversed stress, where the mean stress σ_m is nonzero, the *S-N* diagram approach must be modified. There are four commonly employed design criteria to determine fatigue life by incorporating the damage caused by mean stress.

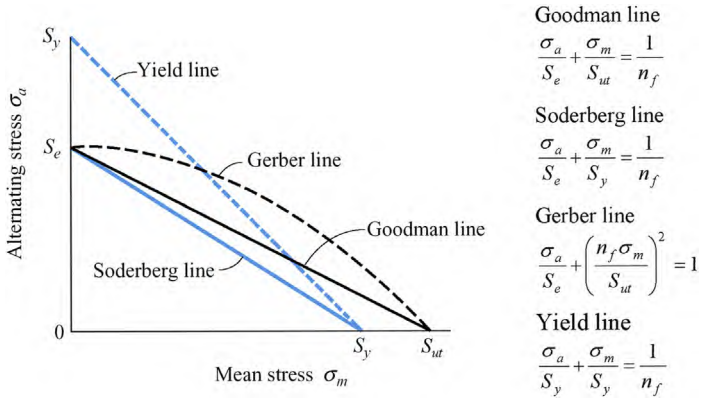


FIGURE 9.6

Fatigue design criteria for nonzero mean stress. (Source: Chapter 7; Bernard J. Hamrock, Steven R. Schmid, Bo O. Jacobson, Fundamentals of Machine Elements, 2nd Ed., McGraw-Hill, 2005.)

They are the Goodman, Soderberg, Gerber, and yield line criteria, as shown in Figure 9.6. Note that, in the figure, if the stress point (alternating stress σ_a and mean stress σ_m) that represents the maximum stress of a component is located below the respective failure line, the component is said to be safe and the fatigue life is infinite (more practically, it is greater than 10^6). The safety factor n_f can be calculated, for example, using the Goodman line criterion as

$$\frac{\sigma_a}{S_e}+\frac{\sigma_m}{S_{ut}}=\frac{1}{n_f} \tag{9.4}$$

Comparing these four criteria, it is obvious that the Soderberg line is the most conservative.

On the other hand, when the stress point is above the failure line, the fatigue life of the component is finite and can be calculated by first converting the mean and alternating stresses into equivalent alternating stress σ_A using one of the design criterion, and then calculating the fatigue life using an *S-N* diagram (i.e., Eq. 9.2a) as a fully-reversed stress case. For example, if using the Goodman line criterion, the equivalent alternating stress is

$$\sigma_A=\frac{S_{ut}\sigma_a}{S_{ut}-\sigma_m} \tag{9.5a}$$

For the Soderberg criterion, the equivalent alternating stress is

$$\sigma_A=\frac{S_y\sigma_a}{S_y-\sigma_m} \tag{9.5b}$$

The modified Goodman line that combines the Goodman line with the yield line provides the best match with experimental data and has been the most popular criterion for fatigue design. In Figure 9.6, the Goodman line and yield line intersect at $\sigma_m=[S_{ut}(S_y-S_e)/(S_{ut}-S_e),\ \sigma_a=S_e(S_{ut}-S_y)/(S_{ut}-S_e)]$. Therefore, when the mean stress σ_m is less than $S_{ut}(S_y-S_e)/(S_{ut}-S_e)$, the Goodman line is employed as the design criterion. If the mean stress is $S_{ut}(S_y-S_e)/(S_{ut}-S_e)<\sigma_m<S_y$, failure is assumed due to yield.

EXAMPLE 9.2

A straight cantilever rotating beam of solid circular cross-section with a 30 mm diameter and 1 m length has an axial load of 50,000 N applied at the end and a stationary transverse (vertical) load of 600 N. The material is AISI 1040 ($S_{ut} = 520$ MPa, $S_y = 320$ MPa). The beam is unnotched.

(a) Find the safety factor for infinite life using the Soderberg line criterion.
(b) What is the equivalent alternating stress σ_A? What is the fatigue life of the beam?
(c) What beam diameter gives a safety factor that exactly equals 1.0?

Solution

First we calculate the endurance limit, alternating stress due to bending, and mean stress due to the axial force. Note that for a more conservative design, we use $S_e = 0.45\ S_{ut}$, although both bending and axial loads are present.

$$S_e = 0.45\ S_{ut} = (0.45 \times 520) = 234 \text{ MPa}$$

$$\sigma_a = \frac{Mc}{I} = \frac{(600)(1000)(30/2)}{\frac{\pi}{64}(30)^4} = 226 \text{ MPa}$$

$$\sigma_m = \frac{P}{A} = \frac{(50{,}000)}{\frac{\pi}{4}(30)^2} = 70.7 \text{ MPa}$$

(a) Safety factor n_f can be obtained using equation of Soderberg line shown in Figure 9.6

$$\frac{\sigma_a}{S_e} + \frac{\sigma_m}{S_y} = \frac{1}{n_f}$$

$$\frac{226}{234} + \frac{70.7}{320} = \frac{1}{n_f} \Rightarrow n_f = 0.840$$

(b) Equivalent alternating stress σ_A

$$\sigma_A = \frac{\sigma_a}{1 - \frac{\sigma_m}{S_y}} = \frac{226}{1 - \frac{70.7}{320}} = 290.1 \text{ MPa}$$

We use the *S-N* approach to calculate fatigue life using the equivalent alternating stress

$$S_{10^3} = 0.75\ S_{ut} = 0.75(520) = 390 \text{ MPa}$$

$$\overline{C} = \log \frac{S_{10^3}^2}{S_e} = \log \frac{390^2}{234} = 2.813$$

$$b_s = -\frac{1}{3} \log \frac{S_{10^3}}{S_e} = -\frac{1}{3} \log \frac{390}{234} = -0.07395$$

$$N = \left(10^{-\overline{c}} \sigma_A\right)^{\frac{1}{b_s}}$$

$$= \left(10^{-2.813} \times 290.1\right)^{\frac{1}{-0.07395}}$$

$$= 55{,}090$$

(c) Beam diameter for $n_f = 1$

$$\frac{\sigma_a}{S_e} + \frac{\sigma_m}{S_y} = 1$$

$$\frac{\frac{32M}{\pi d^3}}{S_e} + \frac{\frac{4P}{\pi d^2}}{S_y} = 1$$

$$\frac{\frac{32(600 \times 1000)}{\pi d^3}}{234} + \frac{\frac{4(50{,}000)}{\pi d^2}}{320} = 1$$

$$\frac{26{,}120}{d^3} + \frac{198.9}{d^2} = 1$$

$$\Rightarrow d = 32 \text{ mm}$$

9.3.3 IN-PHASE BENDING AND TORSION

S-N diagrams and their associated fatigue strengths assume stresses under uniaxial load, either normal stress under axial or bending load or shear stress under torsion load. Most machine element applications encounter more complicated loading conditions. Two special cases are important. The first situation exists where the applied stresses are synchronous and in phase—that is, at the same frequency (synchronous) with a zero phase angle (in phase). This is called simple multiaxial stress. For example, a cylindrical pressure vessel that is periodically pressurized has a hoop and an axial stress that are both directly related to the pressure variation, so they are subject to their maximum and minimum values at the same time. When the forces and torques are not synchronous or in phase, this situation is called complex multiaxial stress, which is discussed in the next subsection.

For simple multiaxial stress, the principal direction of the stress is unchanged. If the stress is fully reversed, an equivalent von Mises stress is calculated as

$$\sigma'_a = \sqrt{\frac{(\sigma_{1_a} - \sigma_{2_a})^2 + (\sigma_{2_a} - \sigma_{3_a})^2 + (\sigma_{3_a} - \sigma_{1_a})^2}{2}} \tag{9.6}$$

where $\sigma_{1_a}, \sigma_{2_a}$, and σ_{3_a} are the alternating principal stresses. The safety factor n_f can thus be calculated using

$$n_f = \frac{S_e}{\sigma'_a} \tag{9.7}$$

for infinite fatigue life. Or, if $\sigma'_a > S_e$, we insert the equivalent stress σ'_a into Eq. 9.2a to estimate fatigue life for $N < 10^6$.

For nonfully reversed stresses, equivalent von Mises stresses are calculated respectively for the alternating stresses (using Eq. 9.6) and mean stress using

$$\sigma'_m = \sqrt{\frac{(\sigma_{1_m} - \sigma_{2_m})^2 + (\sigma_{2_m} - \sigma_{3_m})^2 + (\sigma_{3_m} - \sigma_{1_m})^2}{2}} \tag{9.8}$$

where $\sigma_{1_m}, \sigma_{2_m}$, and σ_{3_m} are the mean principal stresses. Then, design criteria such as the modified Goodman line are employed for safety factor calculation using Eq. 9.4. If the stress point falls outside of the safe region, then Eq. 9.5 may be used to calculate equivalent alternating stress and then to calculate fatigue life N using Eq. 9.2a.

For a more specific load, such as a fully reversed normal stress, caused by either axial force or bending moment ($\sigma_m = 0, \sigma_a \neq 0$) and steady shear stress due to a constant torque ($\tau_m \neq 0, \tau_a = 0$), the safety factor can be obtained by

$$\left(\frac{\sigma_a}{S_e}\right)^2 + \left(\frac{\tau_m}{S_{sy}}\right)^2 = \frac{1}{n_f} \tag{9.9a}$$

where S_{sy} is the modified shear yield strength and $S_{sy} = S_y/\sqrt{3}$. For a fully reversed bending/axial ($\sigma_m = 0$) with fully reversed torque ($\tau_m = 0$), the safety factor can be obtained by

$$\left(\frac{\sigma_a}{S_e}\right)^2 + \left(\frac{\tau_a}{S_{se}}\right)^2 = \frac{1}{n_f} \tag{9.9b}$$

where S_{se} is the endurance limit of shear strength and $S_{se} = S_e/\sqrt{3}$. The relation shown in Eqs 9.9a and 9.9b represents the well-known Gough ellipse, which is backed up by test results (Spotts and Shoup, 1998). Equations 9.9a and 9.9b are derived based on the Soderberg fatigue failure criterion and distortion energy theory. They are especially useful for shaft design.

9.3.4 COMPLEX MULTIAXIAL STRESS

In many practical applications, loads are simultaneously applied in several different directions, producing stresses with no particular direction bias. The principal direction changes from cycle to cycle. These are complex multiaxial stresses or, simply, multiaxial stresses.

Fatigue occurs on the surface where one of the principal stresses is usually zero. On the surface, the stress state is one of plane stress (i.e., a biaxial stress state) so that there are only three stress components σ_x, σ_y, and τ_{xy}. All other stresses are zero. As a result, multiaxial fatigue problems are usually biaxial in nature.

Complex multiaxial stress for fatigue life calculation is still under investigation by many researchers. Many specific cases of complex multiaxial stress have been analyzed, but no overall design approach applicable to all situations has yet been developed. For the common biaxial stress case of combined bending and torsion, such as occurs in shafts, several approaches have been proposed. One of them, SEQA, based on the ASME Boiler Code, is discussed briefly. SEQA is an equivalent or effective stress that combines the normal and shear stresses and the phase relationship between them into an effective-stress value that can be compared to the ductile materials' fatigue and static strength on a modified-Goodman diagram. It is calculated from

$$\text{SEQA} = \frac{\sigma}{\sqrt{2}}\left[1 + \frac{3}{4}Q^2 + \sqrt{1 + \frac{3}{2}Q^2\cos 2\phi + \frac{9}{16}Q^4}\right]^{1/2} \tag{9.10}$$

where

σ is the bending stress amplitude, $Q = 2\tau/\sigma$.
τ is the torsional stress amplitude.
ϕ is the phase angle between bending and torsion.

SEQA can be computed for both mean and alternating stresses. This approach is valid for loads that are synchronous with a predictable phase relationship.

In recent years, criteria based on the critical plane approach for multiaxial fatigue evaluation have been gaining popularity for high-cycle fatigue (You and Lee, 1996). Here, fatigue evaluation is performed on one plane across a critical location in the component. This plane is called the critical plane, which is usually different for different fatigue models. One of the models proposed by Matake (1977) uses a damage parameter based on the linear combination of shear stress amplitude and maximum normal stress acting on the critical plane. The orientation of this plane is described by the spherical coordinates (ϕ_c, θ_c) of its unit normal vector. The critical plane is defined as the plane on which the shear stress amplitude achieves the maximum value:

$$\begin{cases} (\phi_c, \theta_c) = \max_{(\phi,\theta)}[\tau_a(\phi, \theta)] \\ \tau_a(\phi_c, \theta_c) + k\sigma_{\max}(\phi_c, \theta_c) = \xi \end{cases} \tag{9.11}$$

where the subscript c refers to the critical plane. The material constants k and ξ are given as

$$k = \frac{2S_{se}}{S_e} - 1, \quad \text{and} \quad \xi = S_{se} \tag{9.12}$$

Again, S_e and S_{se} are the endurance limits (also called fatigue limits) in fully reversed bending and torsion, respectively. Socie (1993) proposed a stress-based approach for HCF, in which fatigue life is calculated on the critical plane where the total damage is maximum.

$$\tau_a + k_2\sigma_{\max} = \tau_f'(2N)^{b_0} \tag{9.13}$$

The right side of the equation is the elastic part of the strain life. The left side represents the damage parameters defined on the critical plane experiencing the largest range of cyclic shear stress. k_2, τ_f' and b_0 are material parameters.

9.3.5 CUMULATIVE DAMAGE

When a structural component is loaded with multiple stress magnitudes in a set of cyclic loads for their respective number of cycles, the fatigue damage to the component is accumulated from these loading sets. The simplest and most popular approach for calculating cumulative damage is Miner's rule, also referred to as the *Palmgren–Miner linear damage hypothesis*, which states that where there are k different stress magnitudes in a set of cyclic loads, σ_i $(1 \leq i \leq k)$, each contributing $n_i(\sigma_i)$ cycles, then if $N_i(\sigma_i)$ is the number of cycles to failure of the respective stress magnitudes σ_i, failure occurs when

$$\sum_{i=1}^{k} \frac{n_i}{N_i} = C \tag{9.14}$$

where C is experimentally found to be between 0.7 and 2.2. Usually for design purposes, C is assumed to be 1.

Though Miner's rule is a useful approximation in many circumstances, it has major limitations. There is sometimes an effect in the order of the reversals. In some circumstances, cycles of low stress followed by high stress cause more damage than predicted by the rule. This rule does not consider the effect of overload or high stress that may result in a compressive residual stress. High stress followed by low stress may create less damage because of the presence of compressive residual stress.

Although linear damage accumulation has been criticized, it is still the most common approach to fatigue analysis because of its simplicity and effectiveness, and because its accuracy is sufficient in many applications.

EXAMPLE 9.3

A steel bar is under fully reversed bending stresses of $\sigma_{\max} = 300$ MPa and $\sigma_{\min} = -300$ MPa applied for 10,000 cycles. The load is changed to ± 210 MPa and is applied for 20,000 cycles. Finally, the load is changed to ± 350 MPa. How many cycles of operation can be expected at this stress level? Material properties of the part are $S_e = 200$ MPa (modified), $S_y = 490$ MPa, and $S_{ut} = 590$ MPa.

Continued

EXAMPLE 9.3—cont'd

Solution

We first calculate coefficients for the S-N equation based on the given material properties, then use the Miner's rule to calculate the cumulative fatigue life due to the multiple loads.

$$\overline{C} = \log\frac{S_{10^3}}{S_\ell} = \log\frac{(0.9 \times 580)}{200} = 3.15$$

$$b_s = -\frac{1}{3}\log\frac{S_{10^3}}{S_e} = -\frac{1}{3}\log\frac{(0.9 \times 580)}{200} = -0.141$$

$$N_1 = \left(\sigma_1 10^{-\overline{c}}\right)^{\frac{1}{b_s}} = \left(300 \times 10^{-3.15}\right)^{\frac{1}{-0.141}} = 59{,}200$$

$$N_2 = \left(\sigma_2 10^{-\overline{c}}\right)^{\frac{1}{b_s}} = \left(210 \times 10^{-3.15}\right)^{\frac{1}{-0.141}} = 743{,}000$$

$$N_3 = \left(\sigma_3 10^{-\overline{c}}\right)^{\frac{1}{b_s}} = \left(350 \times 10^{-3.15}\right)^{\frac{1}{-0.141}} = 19{,}800$$

Now, use the Miner's rule

$$\frac{n_1}{N_1} + \frac{n_2}{N_2} + \frac{n_3}{N_3} = 1$$

$$n_3 = N_3\left(1 - \frac{n_1}{N_1} - \frac{n_2}{N_2}\right) = 19{,}800\left(1 - \frac{12{,}000}{59,200} - \frac{20{,}000}{743{,}000}\right) = 15{,}900$$

9.4 THE STRAIN-BASED APPROACH

Low-cycle fatigue, where fatigue life N is less than 10^3, is usually attributed to stress that is high enough for local plastic deformation to occur. For plastic deformation, the account in terms of stress is less useful and the strain in the material offers a simpler description. Therefore, the strain-based approach is more widely accepted for life estimates in low-cycle fatigue cases.

9.4.1 THE MANSON–COFFIN EQUATION

Cyclic stress-strain curves are generally obtained from uniaxial stress tests. For a plate with a center hole under normal cyclic loading, as shown in Figure 9.7(a), the material near the crack tip experiences large local stress and strain as shown in Figure 9.7(b). The typical load path of the materials under a complete load cycle can be seen in Figure 9.7(c), where the normal stress is both tensile and compression. In reality cyclic hardening or softening of material alters the shape of the load path. The stress-strain curve shown in Figure 9.7(d) is usually expressed by a Ramberg–Osgood expression (Ramberg and Osgood, 1943):

$$\varepsilon = \frac{\sigma}{E} + \left(\frac{\sigma}{K}\right)^{1/n} \tag{9.15}$$

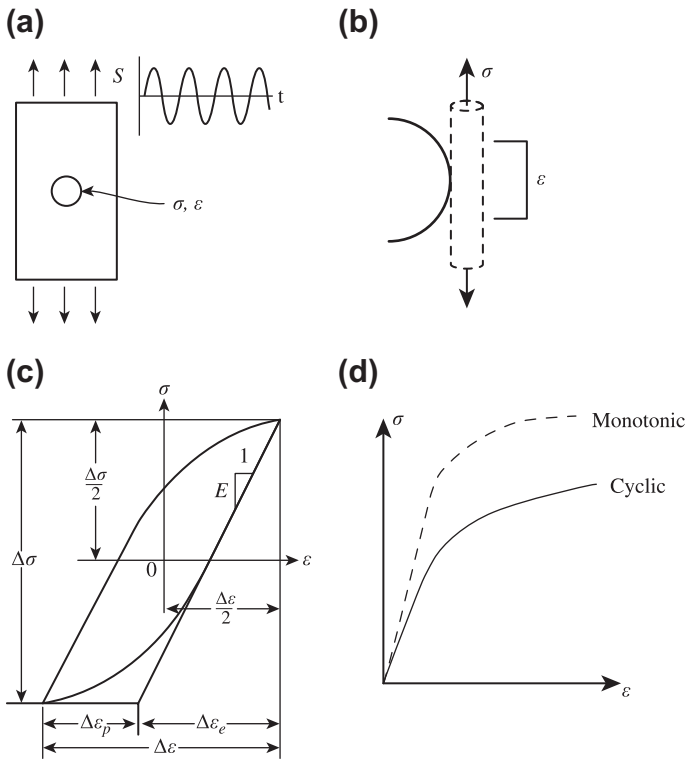


FIGURE 9.7

Strain-based fatigue life prediction: (a) specimen of center crack under cyclic loading, (b) stress and strain around the crack tip, (c) load path in one cycle, and (d) stress-strain curve.

where E is the elastic modulus, and K and n are fitted to the material test data for a particular material. The first term on the right of Eq. 9.15, σ/E, is equal to the elastic part of the strain, while the second term accounts for the plastic part.

As shown in Figure 9.7(d) the stress-strain curves for monotonic and cyclic loads are different. This is because cyclic loading causes relatively larger strain given the same stress. When the strain is monotonic, the slip steps that appear on a crystal surface have a relatively simple topology, as shown in Figure 9.8(a). On the other hand, under cyclic loading, the slip bands tend to group into packets or striations. The surface topology of these striations is more complex and is indicated schematically in Figure 9.8(b). Note that both ridges and crevices tend to be formed. There is good evidence that the crevices are also closely associated with the initiation of cracks.

The first equations proposed for calculating the fatigue life of components correlated the applied stress amplitude with the number of cycles for crack initiation. Basquin's rule (Hertzberg, 1983), presented as Eq. 9.16, was discussed in Section 9.3 (in a slightly different form—for example,

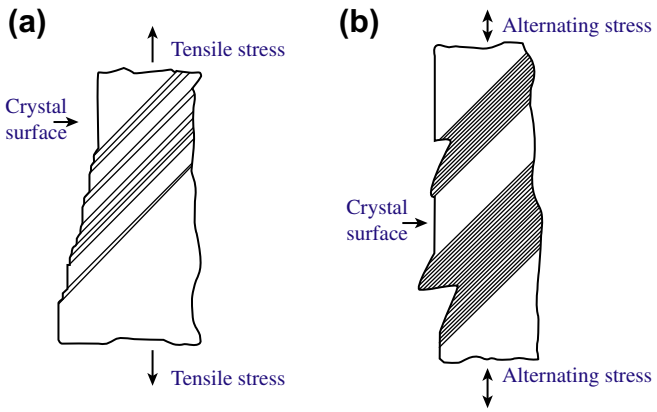


FIGURE 9.8

Slip band observed in elastic material: (a) monotonic loading and (b) cyclic loading.

Eq. 9.3b), where $\sigma_a, \sigma'_f, a,$ and $2N_f$ are, respectively, the stress amplitude, the fatigue strength coefficient, the fatigue strength exponent, and the number of reversals to initiation.

$$\sigma_a = \frac{\Delta\sigma}{2} = \sigma'_f(2N_f)^a \tag{9.16}$$

The relation proposed by Basquin offers good agreement for high-cycle fatigue domains, where elastic deformations are present. For low-cycle fatigue domains, Coffin and Manson (published independently by L.F. Coffin in 1954 and S.S. Manson in 1953) suggest an alternative rule, presented in Eq. 9.17, where $\Delta\varepsilon_p$, ε'_f, and α are, respectively, the plastic strain range in the notch root, the fatigue ductility coefficient, and the fatigue ductility exponent.

$$\frac{\Delta\varepsilon_p}{2} = \varepsilon'_f(2N_f)^\alpha \tag{9.17}$$

The previous two expressions are complementary and can be matched, giving a more general expression valid in any fatigue domain, both high- and low-cycle. The result is represented in Eq. 9.18 as well as in Figure 9.9, where $\Delta\varepsilon$ is the total strain amplitude at the notch root.

$$\frac{\Delta\varepsilon}{2} = \frac{\sigma'_f}{E}(2N_f)^a + \varepsilon'_f(2N_f)^\alpha \tag{9.18}$$

The first and second terms on the right represent the elastic and plastic portions of the deformation, respectively.

Note that Eq. 9.18 assumes fully reversed stress cases. Morrow (1965) modified Eq. 9.18 to take into account the mean stress effect, resulting in Eq. 9.19.

$$\frac{\Delta\varepsilon}{2} = \frac{(\sigma'_f - \sigma_m)}{E}(2N_f)^a + \varepsilon'_f(2N_f)^\alpha \tag{9.19}$$

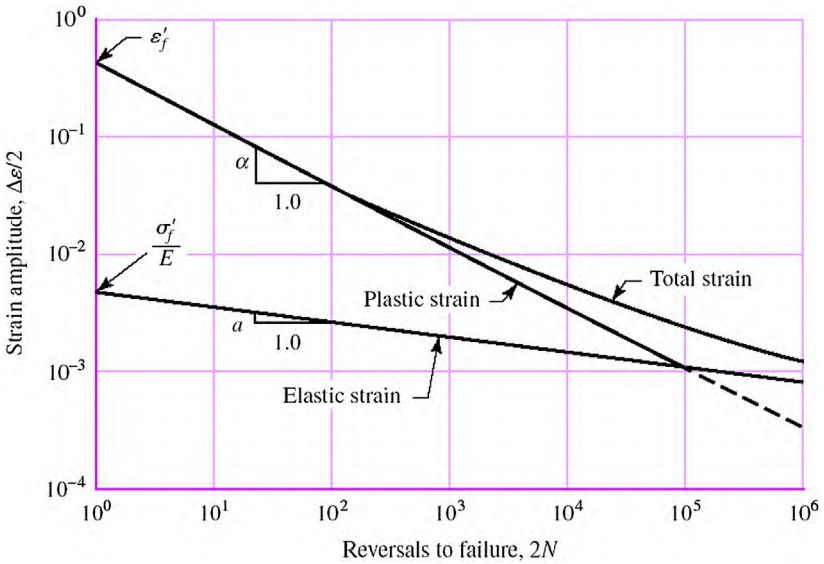


FIGURE 9.9

Plastic and elastic strain-life curve. (Source: From *Shigley 2004*, p. 318.)

Variable-amplitude loads with blocks of constant amplitude can be analyzed using Miner's linear damage accumulation rule, which is similar to that of HCF, as discussed in Section 9.3.5.

Implementing the strain-based approach for fatigue life prediction requires stress and strain calculations in the plastic range. Using the finite element method for stress and strain calculations requires nonlinear plastic analysis, which can be very expensive. Furthermore, the nominal response of the structure is elastic except around a small region at the crack tip. Therefore, a widely accepted approach is to assume a nominally elastic response and then make use of Neuber's (1961) equation to relate local stress and strain to nominal stress and strain at a stress concentration location near the crack tip. If there is no general yielding over the entire structure—that is, if yielding is confined to the local region of the notch—Neuber's rule takes the form

$$\sigma\varepsilon = \frac{(K_t\sigma_0)^2}{E} = \frac{\sigma_e^2}{E} = \sigma_e\varepsilon_e \tag{9.20}$$

The quantities σ and ε are the local stress and strain (in the plastic range) near the crack tip, respectively; σ_0 is a nominal or average stress, and K_t is an elastic stress concentration factor consistently defined with σ_0. After carrying out a linear elastic analysis using the finite element method, stress concentration factor K_t is incorporated, so $\sigma_e = K_t\sigma_0$ in Eq. 9.20. ε_e is the elastic strain.

While the cyclic stress-strain has the form of Eq. 9.15, combining this with Eq. 9.20 gives

$$\sigma_e^2 = \sigma^2 + E\sigma\left(\frac{\sigma}{K}\right)^{1/n} = \sigma^2 + E\sigma^{1+1/n}\left(\frac{1}{K}\right)^{1/n} \tag{9.21a}$$

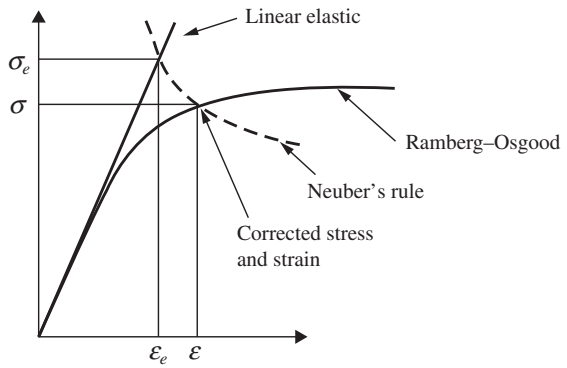


FIGURE 9.10

Stress correction using Neuber's rule.

or

$$\varepsilon^2 = \left(\frac{\sigma_e}{E}\right)^2 + \varepsilon\left(\frac{\sigma_e^2}{E\varepsilon K}\right)^{1/n} = \left(\frac{\sigma_e}{E}\right)^2 + \varepsilon^{1-1/n}\left(\frac{\sigma_e^2}{EK}\right)^{1/n} \tag{9.21b}$$

Thus, for a stress σ_e obtained using linear elastic FEA, Eq. 9.21a or 9.21b can be solved iteratively for the local stress amplitude σ (assuming fully reversed cyclic load) or local strain ε, respectively, as illustrated in Figure 9.10. The half-strain range $\Delta\varepsilon/2 = \varepsilon$ can be used with the strain-life curve, such as in Eq. 9.18, to obtain the estimated number of cycles N_f that arrives at crack initiation.

EXAMPLE 9.4

A notched bar made of steel is under a cyclic tensile load $P = 20{,}000$ lb. The rectangular net cross-section of the bar at the root of the notch is $A = 0.5$ in^2. At the notch the fatigue stress concentration factor is obtained as $K_t = 1.6$. The material properties of the bar that are relevant to our calculation are $K = 155{,}000$ psi, $n = 0.15$, $\varepsilon_f' = 0.48$, $\sigma_f' = 290{,}000$ psi, $a = -0.091$, and $\alpha = -0.60$. How many cycles does it take to initiate a fatigue crack at the notch root?

Solution

The nominal stress σ can be calculated as

$$\sigma = \frac{P}{A} = \frac{20{,}000}{0.5} = 40{,}000 \text{ psi}$$

Thus, the total stress with stress concentration factor K_t is $\sigma_e = 1.6(40{,}000) = 64{,}000$ psi. Using Eq. 9.20,

$$\sigma\varepsilon = \frac{(K_t\sigma_e)^2}{E} = \frac{(1.6 \times 40{,}000)^2}{30 \times 10^6} = 136.5 \text{ psi}$$

From Eq. 9.21b we have

$$\varepsilon^2 = \left(\frac{\sigma_e}{E}\right)^2 + \varepsilon^{1-1/n}\left(\frac{\sigma_e^2}{EK}\right)^{1/n} = \left(\frac{64{,}000}{30 \times 10^6}\right)^2 + \varepsilon^{1-1/0.15}\left(\frac{64{,}000}{30 \times 10^6 \times 155{,}000}\right)^{1/0.15}$$

$$= 4.551 \times 10^{-6} + \varepsilon^{-0.667}(9.176 \times 10^{-8})$$

which can be solved iteratively, or with tools such as MATLAB®, as $\varepsilon = 0.002996$ in./in.

EXAMPLE 9.4—cont'd

Then, from Eq. 9.18,

$$0.002996 = \frac{290{,}000}{30{,}000{,}000}(2N_f)^{0.091} + 0.48(2N_f)^{0.60}$$

which can be solved iteratively as

$$N_f = 6{,}150 \text{ cycles}$$

9.4.2 MULTIAXIAL ANALYSIS

The strain-based approach discussed so far assumes stress and strain under uniaxial loads. As discussed in Section 9.3.3, most machine element applications encounter more complicated loading conditions. Simple multiaxial cases assume that stress or strain is synchronous and in phase. Otherwise, we have complex multiaxial stress and strain.

Similar to the stress-based approach, for simple multiaxial cases the principal direction of the strain is unchanged. An equivalent stress-strain approach is suitable for such cases, such as von Mises equivalent strain or the ASME Boiler Code. The von Mises equivalent strain is further illustrated in this subsection. If the strain is fully reversed, based on the definition of an equivalent stress parameter proposed by the von Mises yield criterion, the equivalent "uniaxial" strain amplitude parameter $\Delta\varepsilon_{\text{eff}}/2$ can be defined as (Chu et al., 1993)

$$\frac{\Delta\varepsilon_{\text{eff}}}{2} = \frac{1}{\sqrt{2(1+\nu_{\text{eff}})}}\left[\left(\frac{\Delta\varepsilon_1}{2}-\frac{\Delta\varepsilon_2}{2}\right)^2+\left(\frac{\Delta\varepsilon_2}{2}-\frac{\Delta\varepsilon_3}{2}\right)^2+\left(\frac{\Delta\varepsilon_1}{2}-\frac{\Delta\varepsilon_3}{2}\right)^2 + 6\left(\frac{\Delta\varepsilon_{12}}{2}+\frac{\Delta\varepsilon_{23}}{2}+\frac{\Delta\varepsilon_{13}}{2}\right)^2\right]^{\frac{1}{2}} \tag{9.22}$$

where

$$\nu_{\text{eff}} = 0.5 - (0.5-\nu)\frac{E_{\text{eff}}}{E} \tag{9.23}$$

where E_{eff} is the effective secant modulus, which is the effective modulus at 2% strain.

As with stress-based approaches, complex multiaxial strain for fatigue life calculation is still under investigation by many researchers. Criteria based on the critical plane approach have gained popularity. Examples include the Fatemi–Socie model (Fatemi and Socie, 1988) and the Smith, Watson, and Topper model (SWT) (Smith et al., 1970).

The Fatemi–Socie model employs the following equation:

$$\gamma_a\left(1+k_1\frac{\sigma_n}{S_y}\right) = \frac{\tau_f'}{G}(2N_f)^a + \gamma_f'(2N_f)^\alpha \tag{9.24}$$

where

γ_a is the shear strain amplitude on the critical plane during a cycle.
σ_n is the maximum normal stress on the critical plane during a cycle.

S_y is the material yield strength.
G is the shear modulus.
k_1, γ_f' and τ_f' are material parameters for fatigue.

The right side of Eq. 9.24 is the description of the strain-life Manson–Coffin curve in torsion. The term on the left side represents the damage parameter on the plane experiencing the largest range of shear strain (critical plane). The fatigue is calculated on the critical plane where the total damage is maximum.

Smith, Watson, and Topper's (SWT) model assumes that the amount of fatigue damage in a cycle is determined by $\sigma_{\max}\varepsilon_a$, where $\sigma_{\max}$ is the maximum tensile stress and ε_a is the strain amplitude on the maximum normal strain plane (critical plane). The fatigue life can be calculated using

$$\varepsilon_a \sigma_{\max} = \frac{\sigma_f'}{E}(2N_f)^a + \sigma_f'\varepsilon_f'(2N_f)^{a+\alpha} \tag{9.25}$$

9.5 FRACTURE MECHANICS*

The mere presence of a crack does not mean that a structure is unsafe. In fact, the damage-tolerant design and analysis approach takes into account the presence of flaws and predicts a component's useful remaining service life (residual life). It is common practice to subject critical structural components to periodic inspections to identify the presence of cracks, and then to monitor crack growth at certain intervals. If the geometric shape of the structure, flaw shape and size, material, and loading, are known, in many cases it is possible to predict the period of subcritical crack growth using crack propagation analysis techniques. Note that a crack undergoing subcritical crack growth propagates under a remotely applied stress (away from the crack tip) well under material yield or ultimate strength.

Crack nucleation can be predicted using the methods discussed in Sections 9.2 and 9.3. The crack grows to a certain size under repetitive or cyclic loads until the stress intensity factor reaches the fracture toughness of the material, at which point the crack grows in an unpredictable manner that leads to component and system failure.

In this section we focus on stage 2 fatigue, crack propagation. The question that is often asked by designers is how the crack grows in a stable fashion, including rate and direction. This question can often be answered by fracture mechanics. We start with a discussion of an infinite plate with a center crack under uniaxial normal load, where both the energy method and the stress intensity factor approaches are touched on. Then we introduce linear elastic fracture mechanics (LEFM), which is the most popular and widely applied method, and follow with a mixed-mode fracture where both normal and shear loads are incorporated. We also briefly discuss the newly developed extended finite element method (XFEM) that successfully alleviates the troublesome mesh generation problem in using regular FEA for crack propagation simulation. In this section, we assume cyclic load. For crack propagation under random loads, peak-valley editing together with rain-flow counting, as to be discussed in Section 9.6, can be employed to convert the random load into a pseudocyclic load. The theory and the computation method introduced are still applicable.

9.5.1 BASIC APPROACHES

There are two basic approaches to fracture analysis: energy criterion and stress intensity. These are equivalent under certain circumstances.

The energy approach states that fracture occurs when the energy available for crack growth is sufficient to overcome the resistance of the material, which is referred to as fracture toughness—an important material property. Griffith (1921) was the first to propose the energy criterion for fracture, but Irwin (1957) is primarily responsible for developing the approach involving energy release rate, G, which is defined as the rate of change in potential energy with crack area for a linear elastic material. When $G = G_c$, the critical energy release rate is reached, which is a measure of fracture toughness.

For a through crack of length $2a$ in an infinite plate subject to a remote tensile stress, as shown in Figure 9.11a, the energy release rate is given by

$$G = \frac{\pi\sigma^2 a}{E} \tag{9.26}$$

where

E is Young's modulus.
σ is the remotely applied stress.
a is the half-crack length.

When G reaches a critical value G_c, due to an increase in stress σ or in crack size a, or both, fracture failure occurs, where

$$G_c = \frac{\pi\sigma_f^2 a_c}{E} \tag{9.27}$$

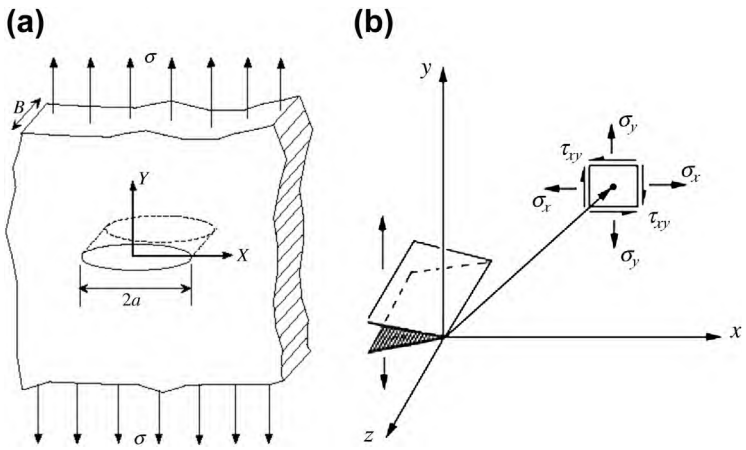


FIGURE 9.11

Infinite plate with center crack of size $2a$: (a) under tensile load and (b) stress element near the crack tip.

The energy release rate G is the driving force for fracture. Note that the remotely applied stress σ_f is usually smaller than the material yield strength (ductile material) or ultimate tensile strength (brittle material). Therefore, whenever cracks appear in the structure, the structure's strength in withstanding operating load deteriorates.

The stress intensity approach employs stress intensity factor (SIF) K_I as the driving force for fracture. For a stress element near the crack tip, the stress components σ_x, σ_y, and τ_{xy}, as shown in Figure 9.11(b), can be calculated using the equations

$$\begin{cases} \sigma_x = \dfrac{K_I}{\sqrt{2\pi r}}\cos\dfrac{\theta}{2}\left(1-\sin\dfrac{\theta}{2}\sin\dfrac{3\theta}{2}\right) \\ \sigma_y = \dfrac{K_I}{\sqrt{2\pi r}}\cos\dfrac{\theta}{2}\left(1-\sin\dfrac{\theta}{2}\sin\dfrac{3\theta}{2}\right) \\ \tau_{xy} = \dfrac{K_I}{\sqrt{2\pi r}}\cos\dfrac{\theta}{2}\sin\dfrac{\theta}{2}\sin\dfrac{3\theta}{2} \end{cases} \tag{9.28}$$

Note that each component is proportional to a single parameter K_I (the SIF). For the plate shown in Figure 9.11a, the SIF is

$$K_I = \sigma\sqrt{\pi a} \tag{9.29a}$$

When the combination of the remotely applied stress σ and the crack size a increases to a critical value K_{Ic}, fracture occurs:

$$K_{Ic} = \sigma_f\sqrt{\pi a_c} \tag{9.29b}$$

Thus, K_{Ic} is an alternate measure of fracture toughness.

Comparing Eqs 9.26 and 9.29a results in a relationship between K_I and G:

$$G = \frac{K_1^2}{E} \tag{9.30}$$

Thus the energy and stress intensity approaches to fracture mechanics are equivalent for linear elastic materials. Note that Eq. 9.30 holds for plane stress problems. For plane strain problems, the denominator on the right becomes $E/(1-\nu)$, where ν is Poisson's ratio.

9.5.2 LINEAR ELASTIC FRACTURE MECHANICS

Linear elastic fracture mechanics is a branch of fracture mechanics that deals with problems in which the size of the plastic zone around the crack tip is very small in comparison to the domain size. LEFM holds well for the brittle mode of fracture, which governs the fracture until the SIFs are greater than the material fracture toughness. Fracture in metals after this point is usually accompanied by significant plastic yielding, and since LEFM is unable to analyze crack growth in such cases, it predicts a conservative estimate of life. However, for a majority of common structural applications, fracture toughness is still considered the failure criterion. Because of this fact, and because of its simplicity, LEFM is widely used.

A major achievement in the theoretical foundation of LEFM was the introduction of SIF K as a parameter for the intensity of stresses close to the crack tip. The SIF shown in Eq. 9.28 is only applicable to the center crack of an infinite thin plate under tensile load, as shown in Figure 9.11a. We must note that the expression for K_I in Eq. 9.28 is different for geometries other than the center-cracked infinite plate. Consequently, it is necessary to introduce a dimensionless geometric factor, Y, to characterize the component geometry, initial crack geometry (e.g., edge crack or center crack), and loads (e.g., tensile, bending, or shear). We thus have

$$K_I = Y\sigma\sqrt{\pi a} \tag{9.31}$$

For example, Y is a function of crack length a and width W of the sheet (under tensile load), given by

$$Y\left(\frac{a}{W}\right) = \sqrt{\sec\left(\frac{\pi a}{W}\right)} \tag{9.32}$$

for a plate of finite width W containing a through-thickness crack of length $2a$ at the center, or

$$Y\left(\frac{a}{W}\right) = 1.12 - \frac{0.41}{\sqrt{\pi}}\frac{a}{W} + \frac{18.7}{\sqrt{\pi}}\left(\frac{a}{W}\right)^2 - \cdots \tag{9.33}$$

for a plate of finite width W containing a through-thickness edge crack of length a.

The geometric factor Y can be obtained analytically using the stress function (as discussed in Chapter 7) for components of simple geometry such as rectangular plates of center or edge cracks, or circular cross-section of a pipe with edge cracks, of which, mostly for planar problems. Note that the famous Westergaard stress function (Anderson, 1994) of complex variables yields Eq. 9.28.

A more general approach for calculating SIF is the J-integral, which is basically a path-independent contour integral around the crack tip. Its use as a fracture parameter was introduced by Rice (1968). The J-integral is given by

$$J = \int_\Gamma \left[W dy - T_i \frac{\partial z_i}{\partial x} ds \right] = \int_\Gamma \left[W\delta_1 - \sigma_{ij}\frac{\partial z_i}{\partial x} \right] n_j d\Gamma \tag{9.34}$$

where

σ_{ij} is the Cauchy stress tensor.
z_i is the ith component of the displacement.
$\boldsymbol{n}$ is the outward normal vector to an arbitrary contour around the crack tip (as shown in Figure 9.12), and $\delta_1\ n_j = n_1$
W is the strain energy, defined as

$$W = \int_0^{\varepsilon_{ij}} \sigma_{ij} d\varepsilon_{ij} \tag{9.35}$$

For linear elastic materials, $W = \frac{1}{2}\sigma_{ij}\varepsilon_{ij}$.

Physically, the J-integral may be interpreted as the energy flowing through the contour Γ per unit crack advance. Under elastic conditions, it is equivalent to Griffith's energy release rate

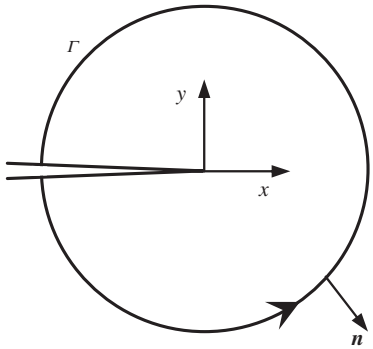


FIGURE 9.12

Path-independent closed contour around the crack tip.

(Anderson, 1994), and its relation to the stress intensity factors for a planar structure in uniaxial load cases is given by

$$G = J = \frac{K_I^2}{E^*} \tag{9.36}$$

where $E^* = E/(1 - \nu^2)$ for plane strain and $E^* = E$ for plane stress. Note that for numerical implementation using the finite element method, the J-integral is usually converted to a domain form by applying the divergence theorem for more accurate results.

9.5.3 MIXED MODE

Up to this point, we have assumed that the structure is under uniaxial loads. In reality, loads are applied to a structural component in different directions. For a structure under general loads, three independent movements of the upper and lower crack surfaces with respect to each other (corresponding to three independent cases of loading) define the three crack opening modes, as shown in Figure 9.13. SIFs for the three modes are denoted by K_I, K_{II}, and K_{III}. In a general case, more than one loading mode may be

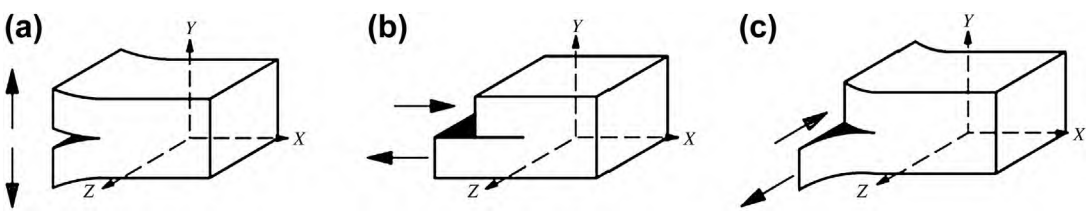


FIGURE 9.13

Modes of crack-tip opening: (a) mode 1: crack opening, (b) mode 2: crack shearing, and (c) mode 3: crack tearing.

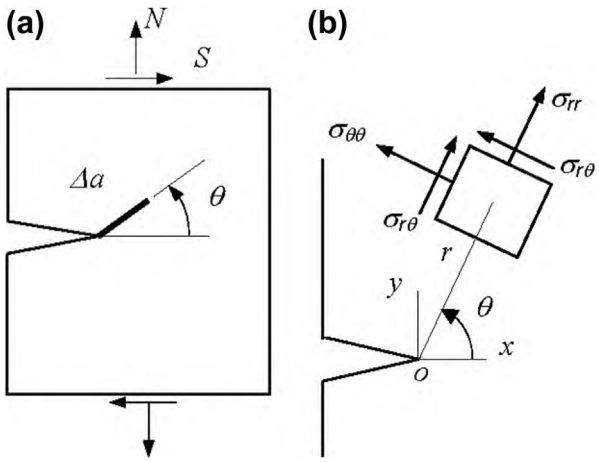


FIGURE 9.14

Mixed mode: (a) normal and shear loads and (b) stress element near the crack tip.

present, and the crack-tip fields can be produced by an appropriate linear combination of those corresponding to these three modes. Such problems are referred to as mixed-mode.

For a planar structural component, mixed-mode problems consist of a crack-opening mode (mode 1) and a shearing mode (mode 2) due to in-plane normal N and shear loads S, respectively, as shown in Figure 9.14a. According to LEFM theory, the displacements and stresses near the crack tip are given, respectively, by (Anderson, 1994)

$$\begin{cases} u_r = \dfrac{1}{\mu}\sqrt{\dfrac{r}{2\mu}}\left[K_I\left(\kappa - \cos\dfrac{\theta}{2} + \sin\theta\sin\dfrac{\theta}{2}\right) + K_{II}\left(\kappa + \sin\dfrac{\theta}{2} + \sin\theta\cos\dfrac{\theta}{2}\right)\right] \\ u_\theta = \dfrac{1}{\mu}\sqrt{\dfrac{r}{2\mu}}\left[K_I\left(\kappa + \sin\dfrac{\theta}{2} + \sin\theta\cos\dfrac{\theta}{2}\right) + K_{II}\left(\kappa - \cos\dfrac{\theta}{2} - \sin\theta\sin\dfrac{\theta}{2}\right)\right] \end{cases} \tag{9.37}$$

and

$$\begin{cases} \sigma_{rr} = \dfrac{1}{\sqrt{2\pi r}}\left[\dfrac{K_I}{4}\left(5\cos\dfrac{\theta}{2} - \cos\dfrac{3\theta}{2}\right) + \dfrac{K_{II}}{4}\left(-5\sin\dfrac{\theta}{2} + 3\sin\dfrac{3\theta}{2}\right)\right] \\ \sigma_{\theta\theta} = \dfrac{1}{\sqrt{2\pi r}}\left[\dfrac{K_I}{4}\left(3\cos\dfrac{\theta}{2} + \cos\dfrac{3\theta}{2}\right) + \dfrac{K_{II}}{4}\left(-3\sin\dfrac{\theta}{2} - 3\sin\dfrac{3\theta}{2}\right)\right] \\ \sigma_{r\theta} = \dfrac{1}{\sqrt{2\pi r}}\left[\dfrac{K_I}{4}\left(\sin\dfrac{\theta}{2} + \sin\dfrac{3\theta}{2}\right) + \dfrac{K_{II}}{4}\left(\cos\dfrac{3\theta}{2} + 3\cos\dfrac{3\theta}{2}\right)\right] \end{cases} \tag{9.38}$$

where K_I and K_{II} are the SIFs of mode 1 and mode 2 crack, respectively. In these expressions, (r, θ) define the location of a point in a local crack tip coordinate system (shown in Figure 9.14b), μ is the shear modulus, ν is Poisson's ratio, and κ is the Kolosov coefficient, whose value is $\kappa = (3 - \nu)/(1 + \nu)$

for plane stress and $\kappa = 3 - 4\nu$ for plane strain. These equations show that the near-tip stresses and displacements are completely determined by the stress intensity factors.

Calculating K_I and K_{II} is not straightforward. No analytical expression is available. The best and most popular way to calculate K_I and K_{II} is to use the J-integral. Again, under elastic conditions, the J-integral is equivalent to Griffith's energy release rate and, for a planar structure of mixed-mode cases, is given by

$$G = J = \frac{1}{E^*}\left(K_I^2 + K_{II}^2\right) \tag{9.39}$$

As shown in Eq. 9.39, the evaluation of the J-integral does not give separate values of mode 1 and mode 2 stress intensity factors. Yau et al. (1980) proposed an interaction integral technique in which two kinematically admissible states of a body are superimposed to extract the mixed-mode SIFs.

Consider two independent equilibrium states of a cracked body. State 1 is defined as the actual state for the given boundary conditions, while State 2 is an auxiliary state. The J-integral for the two superposed states is

$$J^{(1+2)} = J^{(1)} + J^{(2)} + M^{(1+2)} \tag{9.40}$$

where $J^{(1)}$ and $J^{(2)}$ are the J-integrals for actual state and auxiliary state, respectively, and $M^{(1+2)}$ is the interaction integral for the two equilibrium states:

$$M^{(1+2)} = \int_\Gamma \left[W^{(1+2)}\delta_{1j} - \sigma_{ij}^{(1)}\frac{\partial z_i^{(2)}}{\partial x} - \sigma_{ij}^{(2)}\frac{\partial z_i^{(1)}}{\partial x}\right] n_j d\Gamma \tag{9.41}$$

where $W^{(1+2)}$ is the interaction strain energy density:

$$W^{(1+2)} = \frac{1}{2}\left(\sigma_{ij}^{(1)}\varepsilon_{ij}^{(2)} + \sigma_{ij}^{(2)}\varepsilon_{ij}^{(1)}\right) = \sigma_{ij}^{(1)}\varepsilon_{ij}^{(2)} = \sigma_{ij}^{(2)}\varepsilon_{ij}^{(1)} \tag{9.42}$$

On the other hand, Eq. 9.39 for the superposed state can be written as

$$\begin{aligned} J^{(1+2)} &= \frac{1}{E^*}\left[\left(K_I^{(1+2)}\right)^2 + \left(K_{II}^{(1+2)}\right)^2\right] = \frac{1}{E^*}\left[\left(K_I^{(1)} + K_I^{(2)}\right)^2 + \left(K_{II}^{(1)} + K_{II}^{(2)}\right)^2\right] \\ &= J^{(1)} + J^{(2)} + \frac{2}{E^*}\left(K_I^{(1)}K_I^{(2)} + K_{II}^{(1)}K_{II}^{(2)}\right) \end{aligned} \tag{9.43}$$

Thus,

$$M^{(1+2)} = \frac{2}{E^*}\left(K_I^{(1)}K_I^{(2)} + K_{II}^{(1)}K_{II}^{(2)}\right) \tag{9.44}$$

To obtain mode 1 stress intensity factor, the auxiliary state is chosen to be the pure mode 1 asymptotic condition with $K_I^{(2a)} = 1$ and $K_{II}^{(2a)} = 0$. Substituting this in Eq. 9.44 gives

$$K_I^{(1)} = \frac{E^*}{2}M^{(1+2a)} = \frac{E^*}{2}\int_\Gamma \left[W^{(1+2a)}\delta_{1j} - \sigma_{ij}^{(1)}\frac{\partial z_i^{(2a)}}{\partial x} - \sigma_{ij}^{(2a)}\frac{\partial z_i^{(1)}}{\partial x}\right] n_j d\Gamma \tag{9.45}$$

where $z_i^{(2a)}$ and $\sigma_{ij}^{(2a)}$ can be obtained by plugging $K_I^{(2a)} = 1$ and $K_{II}^{(2a)} = 0$ into Eqs 9.37 and 9.38, respectively. Then $\sigma_{ij}^{(2a)}$ and $\varepsilon_{ij}^{(2a)}$ (obtained from $z_i^{(2a)}$) are inserted into Eq. 9.45 for $M^{(1+2a)}$.

Similarly, mode 2 stress intensity factor can be obtained by choosing the auxiliary state to be the pure mode 2 asymptotic condition with $K_I^{(2b)} = 0$ and $K_{II}^{(2b)} = 1$. Substituting this in Eq. 9.44 gives

$$K_{II}^{(1)} = \frac{E^*}{2} M^{(1+2b)} = \int_\Gamma \left[W^{(1+2b)} \delta_{1j} - \sigma_{ij}^{(1)} \frac{\partial z_i^{(2b)}}{\partial x} - \sigma_{ij}^{(2b)} \frac{\partial z_i^{(1)}}{\partial x} \right] n_j d\Gamma \tag{9.46}$$

Note that the asymptotic fields used for the auxiliary state are valid for LEFM only, and hence the interaction integral method presented here is also applicable only to LEFM.

9.5.4 QUASISTATIC CRACK GROWTH

Laboratory experiments show that the rate of crack growth is a function of the stress intensity factor. In Regime B of the crack shown in Figure 9.15 (i.e., the stable crack propagation stage, also called the Paris regime), the crack growth rate can be calculated using the Paris power law (Paris et al., 1961) or Forman's equation (Forman et al., 1967).

The Paris power law relates the crack propagation rate under fatigue loading to stress intensity factors. For a given fatigue loading, assume that $\Delta K = K_{\max} - K_{\min}$ is the SIF range, where $K_{\max}$ and $K_{\min}$ correspond to the SIFs of maximum and minimum stresses in a load cycle,

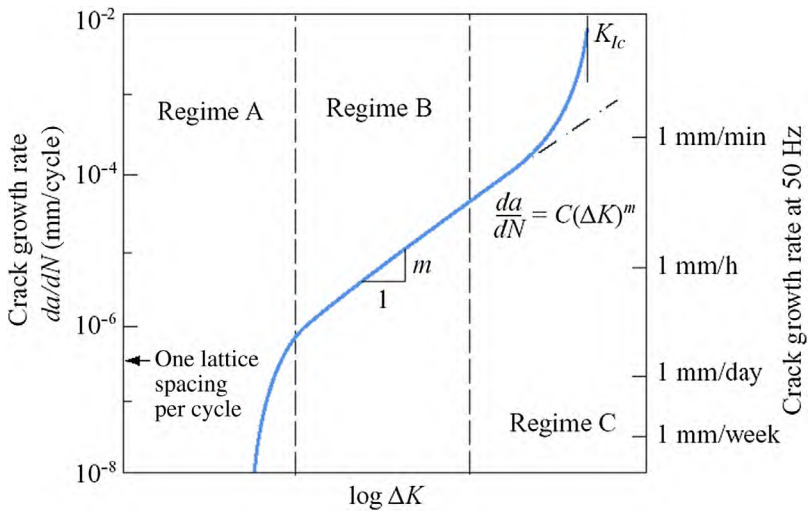


FIGURE 9.15

Rate of crack growth. (Source: Chapter 7; Bernard J. Hamrock, Steven R. Schmid, Bo O. Jacobson, Fundamentals of Machine Elements, 2nd Ed., McGraw-Hill, 2005.)

respectively. Suppose that the crack grows by an amount Δa in ΔN cycles. Crack growth rate is related to ΔK as

$$\frac{\Delta a}{\Delta N} \approx \frac{da}{dN} = C(\Delta K)^m \tag{9.47a}$$

where C and m are material constants. For mode 1 loading, $\Delta K = K_{I\max} - K_{I\min}$. Forman's equation refines the Paris law by incorporating stress ratio R and fracture toughness K_{Ic} into the growth rate calculation:

$$\frac{\Delta a}{\Delta N} \approx \frac{da}{dN} = \frac{C(\Delta K)^p}{K_{Ic}(1-R) - \Delta K} \tag{9.47b}$$

where C, m, and p are material constants for crack propagation.

For general mixed-mode loading, ΔK is replaced by an equivalent SIF range, ΔK_{eq}, given by

$$\Delta K_{\text{eq}} = \sqrt{\Delta K_I^2 + \Delta K_{II}^2} \tag{9.48}$$

Usually, the crack growth per cycle Δa is very small—almost on the order of 10^{-8} in. Hence, in numerical implementation, instead of computing crack growth in every load cycle, usually the value of Δa is predetermined. As a rule of thumb, $\Delta a = a_i/10$, where a_i is the initial crack length. Once Δa is fixed, the number of corresponding cycles is computed using the following equation if the Paris law is employed:

$$\Delta N = \frac{\Delta a}{C\left(\Delta K_{\text{eq}}\right)^m} \tag{9.49}$$

Along with crack propagation rate, crack growth angle θ_c is a necessary parameter in modeling crack propagation. Among the many criteria, the maximum hoop stress criterion is often employed, which states that the crack propagates in a direction where the hoop stress is maximum. Based on the maximum hoop stress criterion, the expression for θ_c is given as

$$\theta_c = 2\tan^{-1}\left(\frac{1}{4}(K_I/K_{II}) \pm \sqrt{(K_I/K_{II})^2 + 8}\right) \tag{9.50}$$

It can be seen from Eqs 9.49 and 9.50 that ΔN and θ_c are functions of K_I and K_{II}.

9.5.5 THE EXTENDED FINITE ELEMENT METHOD

Crack propagation calculations are usually performed using the finite element method. Since the crack tip is singular where the stress field is infinitely high, as shown in Eqs 9.28 or 9.38 according to LEFM theory, a very much refined mesh must be created around it to capture the surrounding high stress, as shown in Figure 9.16a. In addition, when the crack propagates, the structure has to be remeshed to conform to the newly created boundary edges (or faces for 3D) at the crack tip, as illustrated in Figure 9.16b. Very often a large finite element model is required for crack propagation calculation. However, it is more troublesome for mesh, generation around the crack tip for structures with complex geometry, especially for 3D structural components.

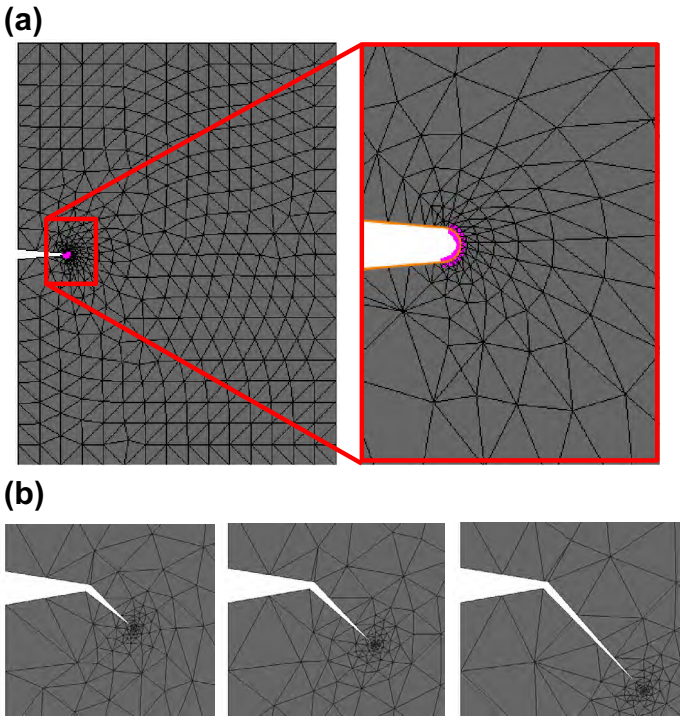


FIGURE 9.16

Finite element mesh for crack growth calculation: (a) very much refined mesh at the crack tip and (b) mesh refinement following the evolving crack tip.

Instead of the regular finite element method, the newly developed extended finite element method alleviates problems of mesh refinement around the crack tip and remesh as the crack grows. XFEM was developed in the late 1990s by Belytschko and Black (1999) to deal with the shortcomings of the finite element method in solving discontinuous problems. One of its key advantages is that in such problems the finite element mesh does not need to be updated to track the crack path or the interface movement.

XFEM is a computational technique in which special enrichment functions are used to incorporate the discontinuity caused by the crack surfaces and crack-tip fields into a regular finite element approximation. The XFEM displacement approximation for a vector-valued function $u(x)$: $\boldsymbol{R}^2 \rightarrow \boldsymbol{R}^2$ is typically given as

$$u^h(\boldsymbol{x},t) = \sum_{i\in I} u_i(t)N_i(\boldsymbol{x}) + \sum_{j\in J} b_j(t)N_i(\boldsymbol{x})H\big(\psi(\boldsymbol{x},t)\big) + \sum_{k\in K} N_k(\boldsymbol{x})\left(\sum_{\ell=1}^{4} a_k^{\ell}(t)B_{\ell}(r,\theta)\right) \tag{9.51}$$

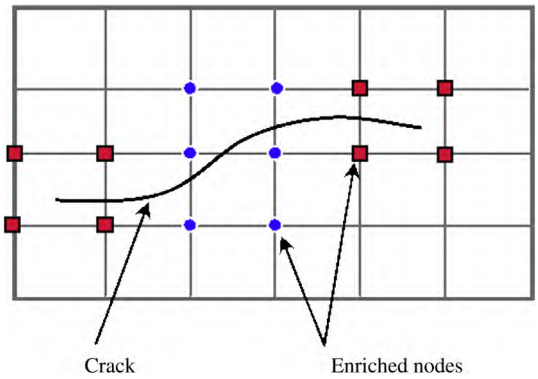


FIGURE 9.17

Nodal enrichment in XFEM.

where

$N_i(\boldsymbol{x})$ is the shape function associated with the node i.
t is a monotonically increasing time parameter that represents the load cycles.
J is the set of all nodes whose support is bisected by the crack (shown by filled circles in Figure 9.17).

The set K contains all nodes of the elements containing the crack tip (shown by filled squares in Figure 9.17).

The first term in Eq. 9.51 is the regular finite element shape function; the second term represents the Heaviside step function $H(\psi(\boldsymbol{x},t))$ employed to model discontinuity due to crack; and the last term incorporates the near-tip asymptotic displacement fields using branch functions, B_ℓ (shown in Figure 9.18), which are defined by

$$B_\ell(r,\theta) = \left\{ \sqrt{r}\sin\frac{\theta}{2}, \sqrt{r}\cos\frac{\theta}{2}, \sqrt{r}\sin\frac{\theta}{2}, \sqrt{r}\cos\frac{\theta}{2}\sin\theta \right\} \tag{9.52}$$

where (r, θ) are defined in a polar coordinate system at the crack tip and $\theta = 0$ is tangent to the crack. Because of the branch functions, a relatively coarse mesh can be used near the crack-tip region.

The level set method (LSM) is a numerical technique used to track the motion of interfaces. In LSM the crack is modeled using signed-distance functions ϕ and ψ (shown in Figure 9.19), which are stored at nodes. Thus, it is always possible to know which elements are cut by the crack and which elements contain the crack tip. LSM couples naturally with XFEM and facilitates selection of nodes for enrichment. These enrichment functions appear in the form of extra degrees of freedom in the finite element stiffness matrix. At the end of a crack growth cycle, the signed-distance functions are updated to account for changes in crack geometry, therefore, no remeshing is required. Thus XFEM and LSM provide an elegant scheme for crack growth simulation.

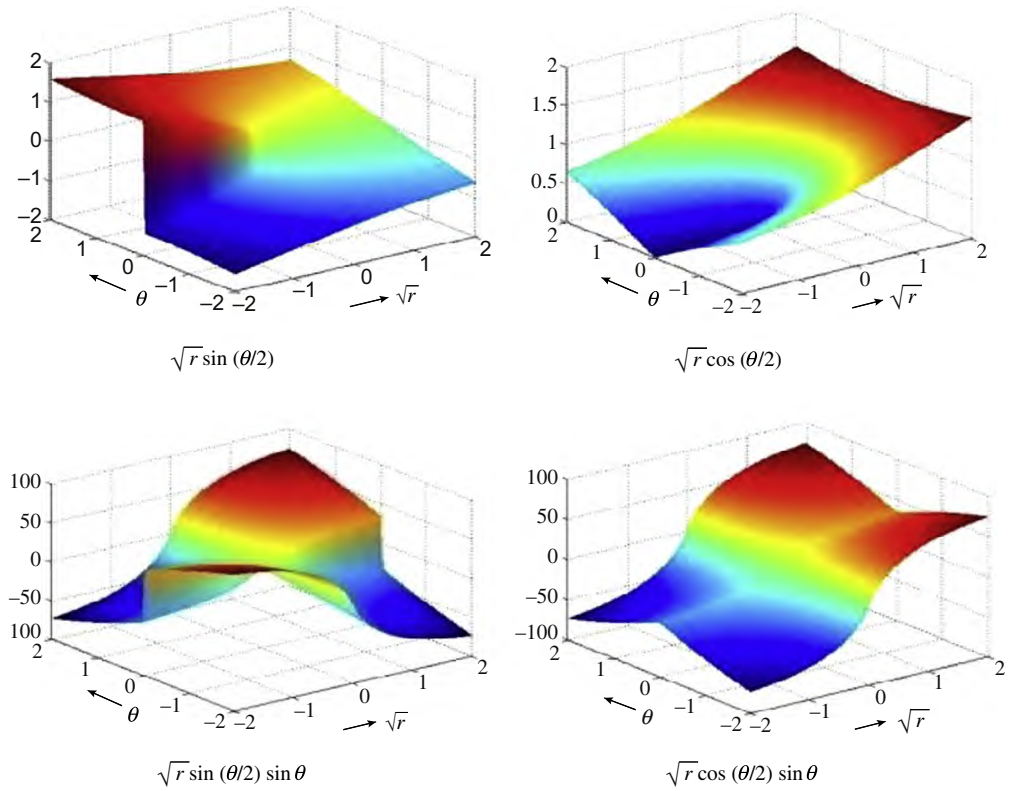


FIGURE 9.18

Branch functions.

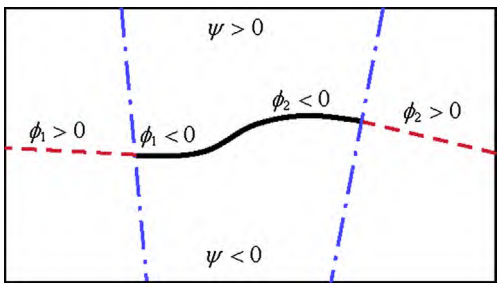


FIGURE 9.19

Crack representation using level sets.

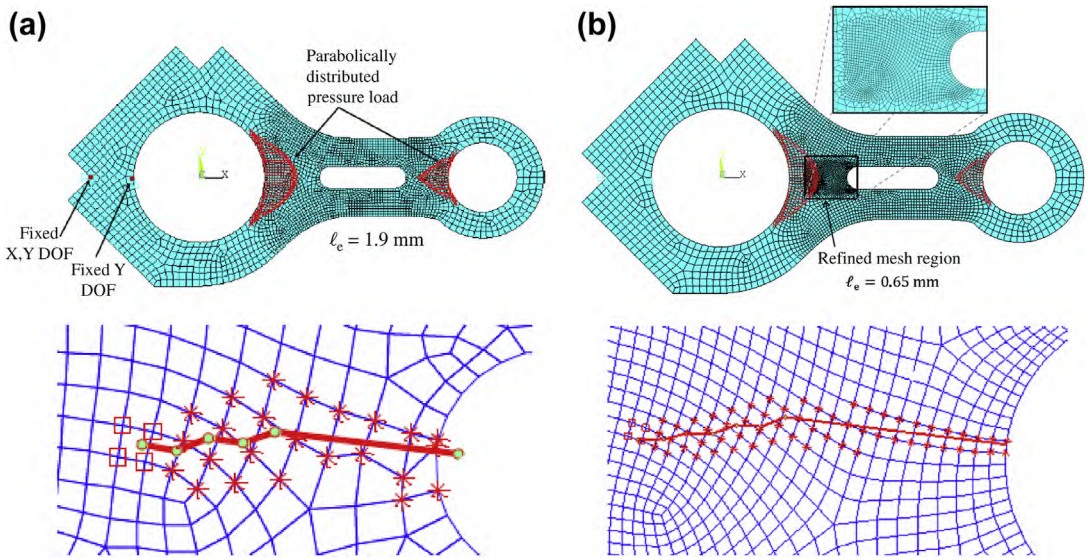


FIGURE 9.20

Two-dimensional engine connecting rod for simulating crack propagation using XFEM and LSM: (a) $\Delta a = 1.5$ mm and 8554 DOF; (b) $\Delta a = 0.8$ mm and 25,186 DOF.

The superiority of XFEM and LSM in solving problems with discontinuities and singularities has been demonstrated and widely recognized. Figure 9.20 demonstrates an application of the method to crack propagation in an engine connecting rod (Edke and Chang, 2011), where the finite element mesh does not have to be updated when the crack tip advances.

In Figure 9.20a, an initial crack of $a = 7$ mm (an arbitrary choice) is introduced at the left side of the semicircular arc of the center slot, as shown. A crack propagation analysis is conducted with a crack growth increment $\Delta a = 1.5$ mm. Figure 9.20a shows the crack propagation path until failure—that is, until the equivalent stress intensity factor K_{eq} exceeds the fracture toughness of the material.

For mixed-mode cases, generally the crack propagation path is curvilinear. In this case, however, the path appears to be zigzag because of the alternating positive and negative signs of the crack propagation angle θ_c. A large crack growth increment and large element size are the main reasons behind these oscillations. The equivalent stress intensity factor K_{eq} exceeds the fracture toughness of the material after four crack propagation cycles, when the final crack size reaches $a_f = 13$ mm ($a + 4\Delta a = 7 + 4 \times 1.5$ mm). When a large value of Δa (such as 1.5 mm) is selected, the analysis may not accurately predict crack path and residual life, as indicated in this coarse mesh case. Note that the estimated residual service life N is about 194,000 load cycles.

To improve analysis accuracy and reduce the zigzag crack path, a smaller Δa must be employed. A very small value of Δa requires very fine mesh, thereby greatly increasing the computational burden. A detailed study was undertaken to examine the effect of mesh refinement and crack growth increments on SIF and crack propagation path. It was found that a refined mesh, shown in Figure 9.20b

with $\Delta a = 0.8$ mm, provides an excellent result, where the crack path is much smoother. The equivalent SIF K_{eq} exceeds the fracture toughness of the material after six crack propagation cycles when the final crack size reaches $a_f = 11.8$ mm ($a + 4\Delta a = 7 + 6 \times 0.8$ mm). The residual service life N is about 129,000 load cycles, which is more conservative than that of the coarse mesh. More details about this example can be found in Section 9.7.2 as a case study.

9.6 DYNAMIC STRESS CALCULATION AND CUMULATIVE DAMAGE

For many applications such as ground vehicles and heavy equipment, the loads applied to heavy load-bearing components are not cyclic. They are random, similar to that shown in Figure 9.4(d). There are two major issues in dealing with such loads for fatigue life calculation. First, calculating stress and strain for every single point of the random loads using the finite element method implies conducting hundreds or thousands of FEAs, which is impractical. Second, the principal directions of the stresses at critical locations of the component vary in time.

The first issue is well addressed using quasistatic FEA for dynamic stress calculations. Various fatigue calculation methods have been proposed to address the second issue (see Sections 9.3 and 9.4). The stress history is then employed to predict the fatigue life of the component using either a strain- or stress-based crack initiation life prediction method and linear elastic fracture mechanics for crack propagation life. The overall process is shown in Figure 9.21.

9.6.1 DYNAMIC STRESS CALCULATIONS

Dynamic stress can be obtained either from experiment (mounting sensors or transducers on a physical component) or from simulation. Using simulation, a representative load history, including inertia forces and external forces (such as joint reaction forces and torques), for accurate dynamic stress computation must first be generated. Multibody dynamic analysis methods (discussed in Chapter 8), which have typically been used for dynamic motion analysis, can be used for dynamic load analysis of

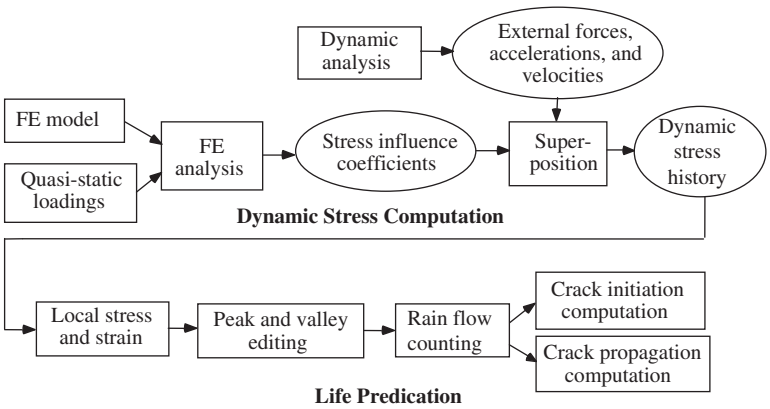


FIGURE 9.21

Computation process for fatigue life.

mechanical systems. All bodies of the dynamic model are usually assumed to be rigid. For the suspension components of a vehicle, the rigid-body assumption usually yields reasonably accurate analysis results to support structural design for durability.

Quasistatic FEA are then performed to obtain stress influence coefficients for the structural components. The stress influence coefficients are superposed with the dynamic analysis results, including external forces, accelerations, and angular velocities to compute dynamic stress history. Sanders and Tesar (1978) showed that the quasistatic deformation evaluation is a valid form of approximation for most industrial mechanisms that are stiff and operate substantially below their natural frequencies. They assumed that deformation caused by applied external and inertia forces is small compared with the geometry of the structural component. They further assumed that the material from which the component is fabricated behaves in a linear elastic fashion.

The finite element model of the component of interest corresponds to a body in the multibody dynamic model. Since dynamic stress histories contain very large amounts of data, it is generally necessary to reduce or condense the amount of data by, for example, peak-valley editing before computing crack initiation and propagation life. These values are then used to perform a cycle-counting procedure to transform variable-amplitude stress or strain histories into a number of constant-amplitude stress or strain histories. These histories are used to compute the component's crack initiation life as well as the crack propagation life.

For a component subject to external forces (including joint reaction forces and torques) and inertia forces obtained from multibody dynamic analysis, the quasistatic equation in a matrix form of the finite element method can be written as

$$\boldsymbol{K}\boldsymbol{z} = \boldsymbol{F}_e(t) - \boldsymbol{F}_i(t) \tag{9.53}$$

where

$\boldsymbol{K}$ is the stiffness matrix.
$\boldsymbol{z}$ is a vector of nodal displacements.
$\boldsymbol{F}_e(t)$ and $\boldsymbol{F}_i(t)$ are vectors of external and inertia force histories, respectively, obtained from dynamic analysis.

Since the loading condition can vary with time in a dynamic system, dynamic stress can be calculated as follows:

$$\boldsymbol{\sigma}(t) = \boldsymbol{D}\boldsymbol{B}\boldsymbol{K}^{-1}[\boldsymbol{F}_e(t) - \boldsymbol{F}_i(t)] \tag{9.54}$$

where

$\boldsymbol{D}$ is the elasticity matrix.
$\boldsymbol{B}$ is the strain-displacement matrix.

The quasistatic method separates the external forces and inertia forces acting on the component into two parts: time dependent (external and inertia force histories) and time independent (quasistatic loading), and treats the quasistatic loading as static forces. The stress influence coefficients are obtained by performing FEA for each quasistatic loading separately. The dynamic stresses can be calculated using the superposition principle; that is, external and inertial force histories are multiplied by the corresponding stress influence coefficients.

A set of unit loads is used to calculate the stress influence coefficients corresponding to the joint reaction forces and torques. The unit loads are applied at a given point $\boldsymbol{x}$ in all degrees of freedom where joint reaction forces and torques act. For example, if a set of joint reaction forces and torques acts on the kth finite element node, the corresponding quasistatic loads $\boldsymbol{q}^k$ are three unit forces and three unit torques in the body reference frame of the body x_1-x_2-x_3 applied to the kth node as six loading cases. Therefore, the stress influence coefficients $\boldsymbol{\sigma}_{SIC}^k$ can be obtained using FEA:

$$\boldsymbol{\sigma}_{SIC}^K = \boldsymbol{DBK}^{-1}\boldsymbol{q}^k \tag{9.55}$$

The inertia body force applied to a point $\boldsymbol{x}$ in the component due to accelerations, angular velocities, and angular accelerations, as shown in Figure 9.22, can be expressed as

$$f_i(\boldsymbol{x}) = f_i^a(\boldsymbol{x}) + f_i^r(\boldsymbol{x}) + f_i^t(\boldsymbol{x}) = -\rho(\boldsymbol{x})a_i - \rho(\boldsymbol{x})a_i^r + \rho(\boldsymbol{x})a_i^t \tag{9.56}$$

where

$\rho(\boldsymbol{x})$ is mass density.
$f_i(\boldsymbol{x})$ is the x_i-component of the inertia body force per unit mass.
$f_i^a(\boldsymbol{x}), f_i^r(\boldsymbol{x})$, and $f_i^t(\boldsymbol{x})$ are inertia body forces per unit mass in the translational, radial, and tangential directions, respectively.
a_i is the instantaneous translational acceleration and is independent of the location of point $\boldsymbol{x}$.
a_i^r is the centripetal acceleration toward the instantaneous axis of the rotation and is perpendicular to it.
a_i^t is the tangential acceleration.

The radial and tangential accelerations $a_i^r(\boldsymbol{x})$ and $a_i^t(\boldsymbol{x})$ at point $\boldsymbol{x}$ can be written as

$$a_i^r(\boldsymbol{x}) = \omega_{ij}\omega_{jk}x_k \tag{9.57}$$

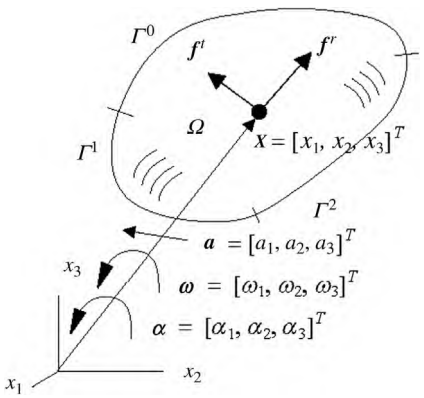


FIGURE 9.22

Inertia forces applied to a component.

and

$$a_i^t(\boldsymbol{x}) = \alpha_{ij} x_j \tag{9.58}$$

where x_k is the kth coordinate of point $\boldsymbol{x}$, ω_{ij} is the instantaneous angular velocity, and α_{ij} is the instantaneous angular acceleration. Hence, the inertia body force at point $\boldsymbol{x}$ is

$$f_i(\boldsymbol{x}) = \rho(\boldsymbol{x})\left(-a_i - \alpha_{ij} x_j + \omega_{ij}\omega_{jk} x_k\right) \tag{9.59}$$

It can be seen from Eq. 9.59 that the inertia force, $f_i(\boldsymbol{x})$, is linearly dependent on components of the acceleration $\boldsymbol{a}$ and the angular acceleration $\boldsymbol{\alpha}$. However, the inertia force is not linearly dependent on components of the angular velocities $\boldsymbol{\omega}$. Instead, it depends linearly on the combinations of components of the angular velocities $\boldsymbol{\omega}$, including six terms $\omega_1\ \omega_2$, $\omega_2\ \omega_3$, $\omega_2\ \omega_3$, $\omega_2^2 + \omega_3^2$, $\omega_1^2 + \omega_3^2$ and $\omega_1^2 + \omega_2^2$. Therefore, to compute the quasistatic loading of inertia forces, 12 loading cases are assumed.

Note that the stress influence coefficients of the first six quasistatic loads can be obtained by applying unit accelerations to perform FEA directly, using commercial FEA codes such as ANSYS®. However, equivalent nodal forces corresponding to the last six quasistatic loads involve angular velocities, which can be applied to the finite element model as external nodal forces. The stress influence coefficients $\boldsymbol{\sigma}_{SIC}^{\text{ine}}$ due to inertia forces can be obtained using FEA:

$$\boldsymbol{\sigma}_{SIC}^{\text{ine}} = \boldsymbol{DBK}^{-1}\boldsymbol{q}_{\boldsymbol{i}}^{\text{ine}}, \quad i = 1, \ldots 12 \tag{9.60}$$

The dynamic stress is calculated using the superposition principle as

$$\boldsymbol{\sigma}(t) = \boldsymbol{\sigma}^{\text{ine}}(t) + \boldsymbol{\sigma}^{\text{ext}}(t) \tag{9.61}$$

where

$$\begin{aligned}\boldsymbol{\sigma}^{\text{ine}}(t) = &\sum_{i=1}^{3} \boldsymbol{\sigma}_{SICi}^{\text{ine}} a_i(t) + \sum_{i=1}^{3} \boldsymbol{\sigma}_{SICi(\text{i}+3)}^{\text{ine}} \alpha_i(t) \\ &+ \boldsymbol{\sigma}_{SIC7}^{\text{ine}} \omega_1(t)\omega_2(t) + \boldsymbol{\sigma}_{SIC8}^{\text{ine}} \omega_2(t)\omega_3(t) + \boldsymbol{\sigma}_{SIC9}^{\text{ine}} \omega_3(t)\omega_1(t) \\ &+ \boldsymbol{\sigma}_{SIC10}^{\text{ine}}\left(\omega_2^2(t) + \omega_3^2(t)\right) + \boldsymbol{\sigma}_{SIC11}^{\text{ine}}\left(\omega_1^2(t) + \omega_3^2(t)\right)\boldsymbol{\sigma}_{SIC12}^{\text{ine}}\left(\omega_1^2(t) + \omega_2^2(t)\right)\end{aligned} \tag{9.62}$$

in which $\boldsymbol{\sigma}_{SIC}^{\text{ine}}$ is obtained from Eq. 9.60, and

$$\boldsymbol{\sigma}^{\text{ext}}(t) = \sum_{k=1}^{n} \boldsymbol{\sigma}_{SIC}^{k} \boldsymbol{F}_i^k(t) \tag{9.63}$$

where $\boldsymbol{\sigma}_{SIC}^{k}$ can be obtained from Eq. 9.55, and n is the number of nodes at which external forces $\boldsymbol{F}^k(t)$ are applied.

9.6.2 PEAK-VALLEY EDITING

As discussed earlier, variable-amplitude fatigue life prediction usually requires either strain or stress histories as input. Often these strain or stress histories are in the form of time histories obtained from multibody dynamic analysis (or from experiment by mounting sensors or transducers on a physical component for data collection). These time histories contain stress or strain values collected or

computed at preset time intervals. Depending on the time step used and the length of the analysis, these types of stress-time or strain-time histories can often contain very large amounts of data. For the purpose of fatigue life computation, it is generally necessary to reduce or condense the amount of data in them. The cycle-counting process, such as rain-flow counting, requires that a variable-amplitude time history be put into reduced form (peak-valley edited) before life calculations can be performed. Also, fewer data are easier to manipulate.

A common method for reducing the number of points in a variable-amplitude time history is to perform peak-valley editing on it (Downing and Socie, 1982). Peak-valley editing has two main purposes. The first is to produce a history that contains only sequential peak and valley reversal points. Peaks and valleys are points in the time history where a change in loading direction occurs. For example, data between points A and B in Figure 9.23(a) do not contain any stress reversals; hence, they are all removed. The second purpose is to remove any ranges with magnitudes less than a prescribed minimum allowable value. For example, if the minimum range of stress variation is 20 ksi, all data points between points B and C are removed, as shown in Figure 9.23(b).

9.6.3 RAIN-FLOW COUNTING

Most traditional fatigue life prediction models require a constant-amplitude loading as input. Constant-amplitude loading is usually represented by repeating sinusoidal or triangular waveforms. In the case of fatigue crack initiation prediction, the conventional strain-life relationship is used to relate a constant-amplitude strain loading $\Delta\varepsilon/2$, shown in Figure 9.24, to the fatigue crack initiation life $2N_f$, using, for example, Eq. 9.18.

However, components of mechanical systems usually experience a variable-amplitude loading that yields a stress or strain history with variable amplitude, such as the one shown in Figure 9.23. To predict fatigue life of a component subject to a variable-amplitude loading history, the variable-amplitude stress or strain history must be converted into several constant-amplitude cycles. The procedure for this conversion is referred to as the cycle counting procedure. A number of such procedures have been proposed over the years, but all of them must use some rules to decide when or how a cycle is defined from a variable-amplitude history. A well-accepted procedure, called rain-flow counting (Downing and Socie, 1982), is discussed next. This procedure attempts to define cycles that correspond to a closed stress-strain hysteresis loop.

The rain-flow counting algorithm is used in the analysis of fatigue data to reduce a spectrum of varying stress into a set of simple stress reversals. Its importance is that it allows the application of Miner's rule to assess the fatigue life of a structure subject to complex loading. The algorithm was developed by Endo and Matsuiski in 1968; they describe the process in terms of rain falling off a pagoda roof.

Figure 9.25(a) shows a typical stress history, composed of repetitive blocks. A stress block (stress points A to G) is identified and rotated 90 deg. clockwise, as shown at the top of Figure 9.25(b). The rotated stress block (or loading history) (points A–G) resembles a Japanese pagoda. The corresponding stress and strain history is plotted directly below the loading history. In the lower stress-strain plot, three cycles are easily identified: one large overall cycle (A–D–G), one intermediate cycle in the center of the plot (C–B–C), and one smaller cycle (E–F–E). Each cycle has its own strain range and mean stress.

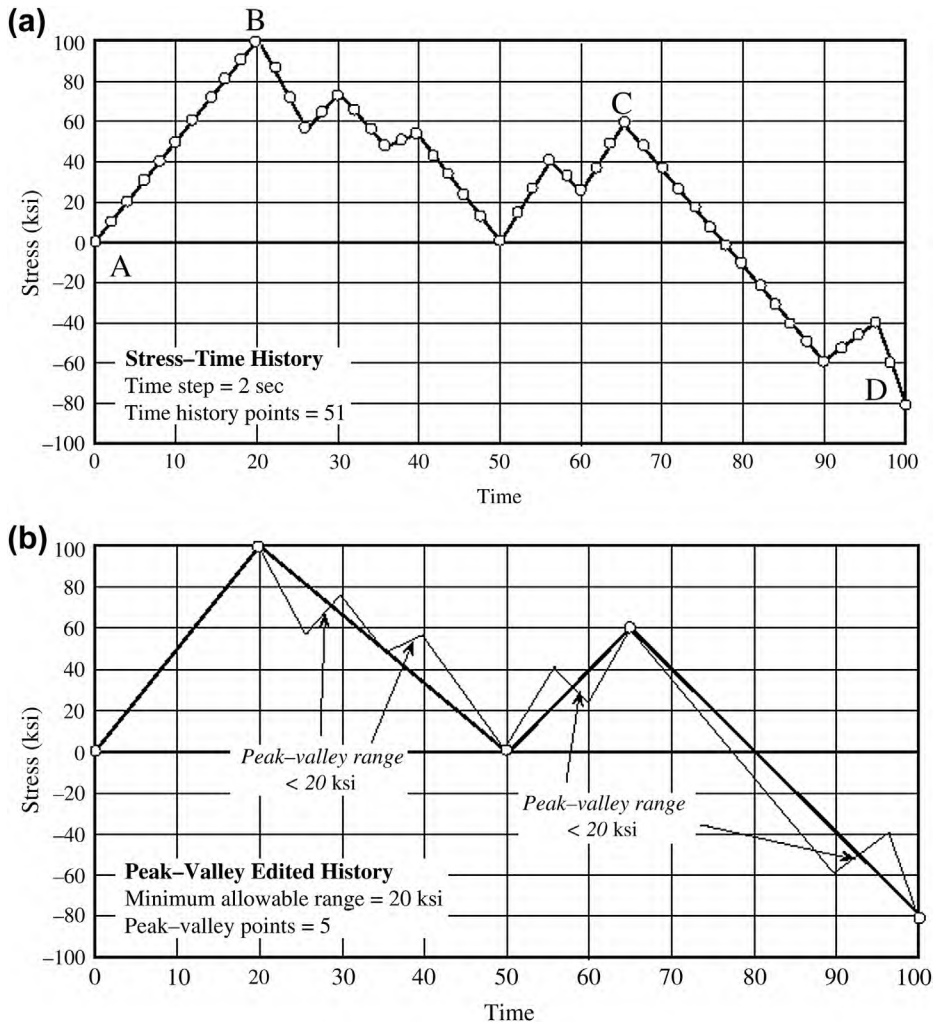


FIGURE 9.23

Peak-valley editing: (a) original stress-time history and (b) edited stress-time history.

From a deformation viewpoint, the process proceeds as follows. Start at A, the maximum strain, and load the material to B in compression. Then reload to point C and compress to D. When the material reaches the strain at point B during the loading from C to D, the material remembers its prior deformation and deforms along a path from A to D as if event C–B never happened. This is better illustrated in the next part of the loading. Load from D to E and unload to F. Now load from F to G. When the material reaches the strain at point E during the loading from F to G, the material remembers its prior deformation and deforms along a

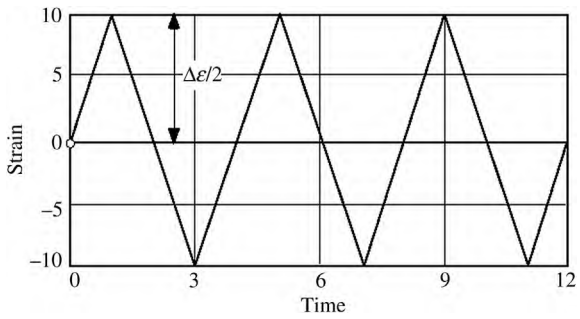


FIGURE 9.24

Constant-amplitude strain history.

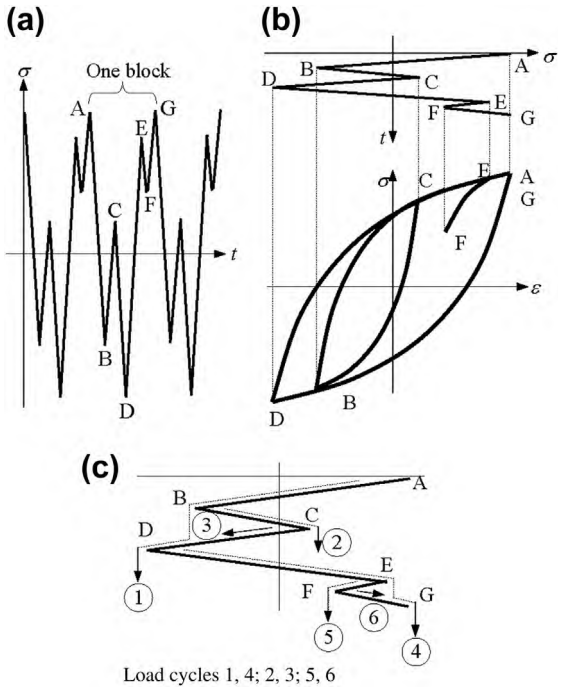


FIGURE 9.25

Rain-flow cycle counting: (a) stress history, (b) hysteresis loops, and (c) rain-flow counting.

path from D to G as if the event E–F never happened. As a result, rain-flow counting identifies three cycles: A–D–G, B–C–B, and E–F–E.

Several rules are imposed on raindrops falling down these sloping roofs so that the rain flow may be used to define cycles and half-cycles of fluctuating stress in the spectrum. Rain flow is initiated by

placing raindrops successively at the inside of each peak (maximum) and valley (minimum). The rules are as follows:

1. The rain is allowed to fall on the roof and drip down to the next slope. For example, the rain flow begins at point A, drips to point B, then to point D. After point D, there are no more roofs to drip to, so stress points A–D are counted as a half-cycle, cycle 1.
2. Step 1 is true except that, if the rain drop initiates at a valley (for example, point B shown in Figure 9.25(c)), it must be terminated when it comes opposite a valley more negative (in this case point D) than the valley from which it initiated. Therefore, stress points B–C are counted as a half-cycle, cycle 2.
3. Similarly, if the rain flow initiates at a peak, it must be terminated when it comes opposite a peak more positive than the peak from which it initiated.
4. The rain flow must stop if it meets the rain from a roof above. For example, the rain begins at point C and ends beneath cycle 1, right under point B. Therefore, stress points C–B are counted as a half-cycle, cycle 3.
5. Half-cycles of identical magnitude (but opposite sense) must be paired up to count the number of complete cycles.

Following the rules, from Figure 9.25(c), stress points D–E–G are counted as a half-cycle (rule 1), cycle 4. Stress points E–F are counted as half-cycle 5 (rule 1), and points F–E are counted as half-cycle 6, as shown in Figure 9.25(c). Half-cycles 1 and 4 form a complete cycle (rule 5); hence, the hysteresis loop A–D–G shown in Figure 9.25(b). Similarly, half-cycles 2 and 3 form the second cycle, the smaller loop B–C in Figure 9.25(b). Finally, half-cycles 5 and 6 form the third cycle, the smallest loop F–G in Figure 9.25(b).

9.6.4 BLOCKS TO FAILURE

Obtaining the fatigue crack initiation life $2N_f$ for each rain-flow cycle does not directly yield the required "blocks to failure" result for the fatigue crack initiation life prediction involved with the variable stress or strain amplitudes. These individual lives must be combined through the use of an appropriate damage summation routine to obtain the required "blocks to failure" result.

The most popular and widely used damage summation is the Palmgren–Miner linear damage rule, or Miner's rule, discussed in Section 9.2.5. Miner's rule simply states that the failure occurs when the summation of the individual damage values caused by each rain-flow cycle reaches a value of 1; that is,

$$\sum_{i=1}^{k} D_i \geq 1 \tag{9.64}$$

where

k is the total number of cycles defined from the variable-amplitude history.
D_i is the damage for the ith defined cycle.

Miner's rule defines the damage per individual rain-flow cycle as

$$D_i = \frac{2}{(2N_f)_i} = \frac{1}{(N_f)_i} \tag{9.65}$$

where

$(2N_f)_i$ is the fatigue crack initiation life (reversals to failure) for the ith defined cycle obtained by solving the nonlinear strain-life relation, Eq. 9.18 or 9.19.

From the definition of damage per cycle given in Eq. 9.65, it can be seen that a given rain-flow cycle consumes $1/(N_f)_i$ of the total fatigue, where $(N_f)_i$ is the fatigue life in cycles to failure for the given ith cycle.

Defining the total accumulated damage for one loading block as D_{block}, Miner's rule predicts failure when

$$D_{\text{block}} \geq 1 \tag{9.66}$$

Using Eqs 9.65 and 9.66, Miner's rule can be expressed in a form that directly yields the variable-amplitude fatigue life in "blocks to failure" as

$$B_f = \frac{1}{D_{\text{block}}} = \frac{1}{\sum_{i=1}^{k} D_i} = \frac{1}{\sum_{i=1}^{k} \frac{2}{(2N_f)_i}} \tag{9.67}$$

where B_f represents the blocks to failure due to the given variable-amplitude loading.

This can be thought of as assessing what proportion of life is consumed by stress reversal at each magnitude and then forming a linear combination of their cumulative effects.

9.7 FATIGUE AND FRACTURE SIMULATION SOFTWARE

Three types of fatigue software are included in this section: the general-purpose codes for crack initiation, non-FEA crack propagation, and FEA crack propagation.

9.7.1 GENERAL-PURPOSE CODES FOR CRACK INITIATION

Several general-purpose codes have been developed for predicting crack initiation life, supporting both high- and low-cycle fatigue. Some support basic stress-life and strain-life estimates of uniaxial stresses. These include the fatigue computation of Pro/MECHANICA Structure (www.ptc.com) and SolidWorks Simulation (www.solidworks.com). Some codes are implemented as postprocessors or modules of commercial FEA software, including ANSYS (www.ansys.com), MSC Fatigue® (www.mscsoftware.com), and ABAQUS® (www.3ds.com).

nCode DesignLife™ (www.ncodeinc.com), an ANSYS module, provides advanced fatigue analysis in the ANSYS Workbench 11 SP1 environment. Results and materials data from Workbench simulations can be directly accessed by DesignLife, which was developed by HBM-nCode, Inc. Both stress-life (S-N) and strain-life (ε-N) fatigue life estimations are implemented in this program. It also implements both Goodman and Gerber mean stress correction. In strain-life calculations, DesignLife uses Neuber (1961) and Hoffmann and Seeger (1985) notch corrections and mean stress correction by Morrow (1965) or Smith-Watson-Topper (1970). When temperature difference is important in a model, the software provides difference curves for different temperatures and interpolates between them. For multiaxial problems, the Dang Van (Dang Van and Papadopoulos, 1987; Dang Van et al., 1989) method is used. In addition, the software has spot and seam weld tools and allows temperature variations to be included in simulations.

MSC.Fatigue is an advanced fatigue life estimation software package for use with finite element analysis results. It provides state-of-the-art fatigue design tools that can be used to optimize product life. MSC.Fatigue was developed by nCode International Ltd. in conjunction with MSC Software Corporation. It is integrated with MSC/Patran® for its excellent meshing and visualization capabilities, as well as with MSC Adams® for dynamic simulation and dynamic stress calculations for fatigue life estimates.

Safe Technology Ltd. has developed a set of durability calculation programs collected under the name fe-safe™ (www.safetechnology.com). In normal stress-life fatigue calculation, fe-safe implements Goodman (Dowling, 2007), Gerber (Dowling, 2007), Buch (1997), or user-defined mean stress corrections. Morrow (1965), SWT (1970), or user-defined mean stress correction is available for estimating life with the strain-life method. For multiaxial analysis the software gives the option of Dang Van (Dang Van and Papadopoulous, 1987; Dang Van et al., 1989) for stress-based analysis and Brown-Miller (Brown and Miller, 1973) for combining normal and shear strain in strain-based analysis. All of these modules are collected in a standalone product called safe4-fatigue™ (www.safetechnology.com), a suite that can be used for fatigue calculation. fe-safe has been integrated into ABAQUS as a postprocessor for fatigue life estimates.

FEMFAT (www.femfat.com) was created by the Engineering Center Steyr in Austria. The software is a fatigue postprocessor utilizing finite element method to predict fatigue life in components. Finite element modeling is carried out in a separate program, and FEMFAT is compatible with many of the most commonly used CAE programs.

In addition to commercial-based fatigue programs, there are a number of online free fatigue postprocessors. One of these was developed by Darrell F. Socie, a professor of mechanical engineering at the University of Illinois, and is called Fatigue Calculator. The software is opened in a web browser with no need for installation.

9.7.2 NON-FEA-BASED CRACK PROPAGATION

To date, there are two mainstream approaches available for the calculation of fracture parameters—the closed-form solution and the finite element method (including the boundary element method). In the closed-form approach a known solution provides fracture parameters as a function of geometry, external loads, and crack size. The known solutions may be derived from analytical expressions or more usually, interpolated from results obtained using FEM. These predictions generally assume that a crack grows under a single prescribed mode, mostly mode 1 (crack opening) conditions in a single plane (normal to the principal stress direction for some limiting load conditions). As a result of these simplifications, a constant principal stress orientation must be assumed. However, if it can be used, this approach provides a potentially fast method to determine crack propagation. A number of software packages use this method—for example, AFGROW (www.afgrow.net) and NASGRO® (www.swri.org)—because it is efficient, but it is limited since it does not support structures of complex geometry and general loading conditions.

AFGROW is a fatigue analysis software program dedicated to crack propagation calculation. Its development began in the early 1980s under the name ASDGRO. Since then it has been in constant development and has been rewritten multiple times. It was first used in crack growth analysis for the Sikorsky H-53 helicopter. AFGROW is currently being further developed and is being used by the U.S. Air Force.

For crack growth, AFGROW can calculate a single crack, multiple cracks, and single or multiple asymmetric cracks. Also, it takes residual stresses into account, and contains multiple crack growth laws for the user to choose from. In addition to pure crack growth AFGROW allows inclusion of temperature, retardation, and repair (if the crack has a bonded repair patch) effects on the crack. To support these calculations, a library with material data is included that can be modified, and in which new materials can be added or old ones edited. The methods for calculating crack growth are the Forman equation (Forman et al., 1967), the Harter T-method (Harter, 1994), the NASGRO equation (Forman and Mettu, 1992), the Walker equation (Walker, 1970), and tabular lookup, which follows the Harter T-method and the Walker equation, interpolating between curves for different stress ratios.

NASGRO is a crack propagation program with roots in the 1980s. It was created by NASA for fracture control analysis of space hardware. It was further developed by NASA in collaboration with SwRI (Southwest Research Institute) for aging aircraft, and the software was expanded for use in the broader aerospace/aircraft industry. NASGRO is entirely a crack growth program. The equation is a further development of the Paris law to give better crack growth prediction. Because of this equation, many new constants, depending on the material, are required and it is therefore implemented in a large database of material data. Addition of new user-specified material data is also possible. When loading the component, the software uses spectrums of load data called blocks. Several blocks can be added together to create the total load. NASGRO was developed around its own crack growth law, which is a more complex derivation of the Paris law or, more specifically, an improvement of it, by Forman and Mettu (1992).

9.7.3 FEA-BASED CRACK PROPAGATION

The primary advantage of FEM is its flexibility in terms of types of analysis (such as nonlinear), geometry complexity, and general loading conditions. Many commercial FEM packages are available for fracture mechanics applications. To support crack propagation, add-in software modules such as ZENCRACK (www.zentech.co.uk) provide commercial FEM packages with a 3D crack propagation capability, including ABAQUS MSC.Marc® and ANSYS. ZENCRACK uses the *J*-integral for calculating fracture parameters. A similar approach is employed in Franc2D and Franc3D (www.cfg.cornell.edu/index.htm), which use displacements to calculate stress intensity factors and crack growth. Also, MSC.Fatigue incorporates nCode®, developed by nCode International Ltd. (Fjeldstad and Wormsen, 2006) for fracture analysis. As for boundary element methods, BEASY™ (www.beasy.com) is the best known commercial code with crack propagation capabilities. It provides a linear solution with *J*-integral capabilities for 2D applications. In 3D the linear solution uses displacements to generate stress intensity factors.

All of these tools require regenerating the finite element mesh as the crack advances, in addition to creating very refined mesh around the crack tip to capture high stress in the area. This requirement leads to extremely large finite element models and requires extensive computation time. Essentially, a successful crack growth simulation is at the mercy of the automatic mesh generator offered by these tools.

The newly developed XFEM with LSM alleviates the mesh problem. The first version of XFEM was recently implemented in ABAQUS 6.9.

9.8 CASE STUDIES AND TUTORIAL EXAMPLE

Two case studies, a tracked vehicle roadarm for crack initiation and 2D engine connecting rod for crack propagation using XFEM, are included in this section. In addition, one tutorial example is presented.

9.8.1 CASE STUDY: TRACKED VEHICLE ROADARM

A roadarm of a tracked vehicle shown in Figure 9.26 (Chang et al., 1997) is employed as a case study to demonstrate the crack initiation life prediction discussed in this chapter. The multibody dynamic model of the tracked vehicle and its simulation environment are described first. Then a structural finite

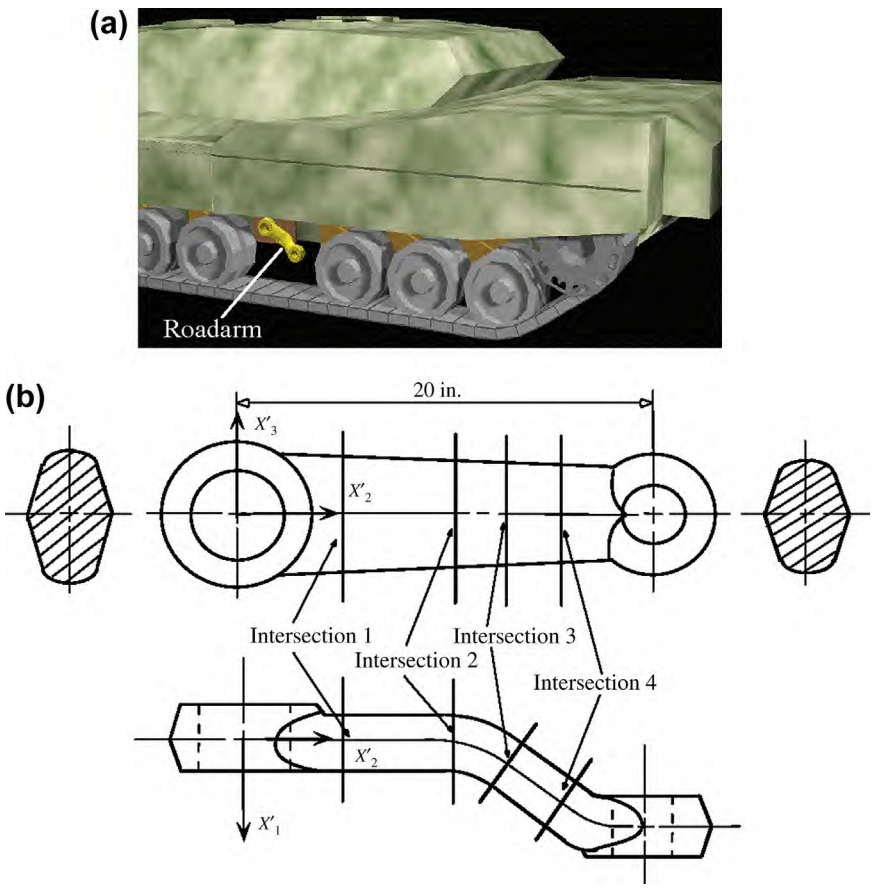


FIGURE 9.26

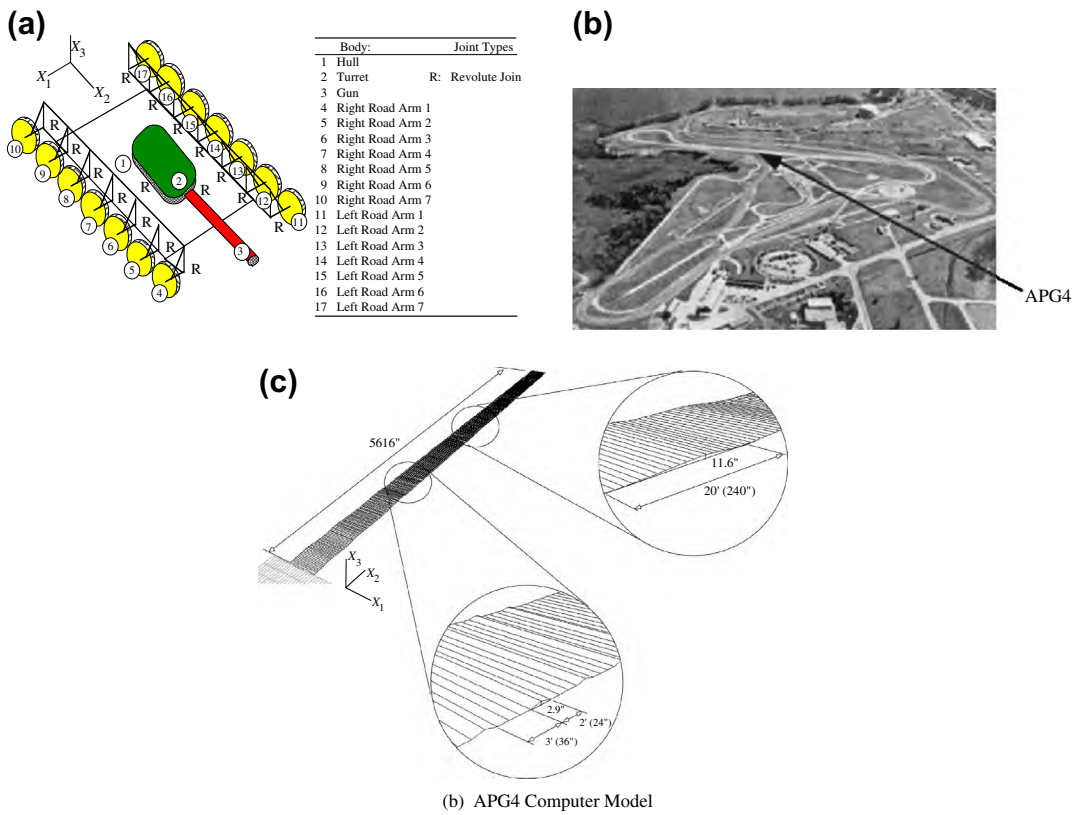
(a) Tracked vehicle; (b) roadarm geometric model.

element model of the roadarm is presented with contours of crack initiation life and von Mises stress at the peak load of the simulation period.

A 17-body dynamics model, shown in Figure 9.27(a), was generated to carry out vehicle dynamic analysis for the tracked vehicle on Aberdeen Proving Ground 4 (APG4), as shown in Figure 9.27(b), at a constant speed of 20 miles per hour forward (positive X_2-direction in Figure 9.27(a)). The road was fairly uneven, featuring a few bumps and washboard-like terrain, as shown in Figure 9.27(c).

Dynamic analysis and design systems (DADS) were used to generate the dynamics model and to perform dynamic analysis. A 20-sec dynamic simulation was performed at a maximum integration time step of 0.05 sec. An output interval of 0.05 sec was predefined for this analysis, and a total of 400 sets of results were generated. The joint reaction forces applied at the wheel end of the roadarm, accelerations, angular velocities, and angular accelerations of the roadarm were obtained from the analysis. A time history of joint reaction forces at the wheel end is shown in Figure 9.28.

Four beam elements, STIF4, and 310 ANSYS 20-node isoparametric finite elements, STIF95, were used for the roadarm FEM model. A number of rigid beams were created to connect nodes at the inner



(b) APG4 Computer Model

FIGURE 9.27

Dynamic simulation: (a) Tracked vehicle dynamic model, (b) APG4, and (c) APG4 road profile.

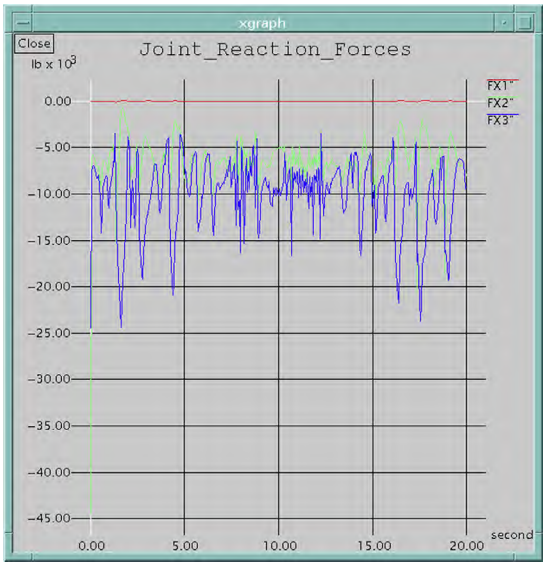


FIGURE 9.28

Joint reaction forces applied to the roadarm.

surface of the two holes to end nodes of beam elements to simulate the roadwheel shaft and torsion bar, respectively. Displacement constraints were defined at the end nodes of the beam elements that simulated the torsion bar, and joint reaction forces and torque were applied at the end node of the other beam element that simulated the shaft of the roadwheel, as shown in Figure 9.29. The roadarm was made of S4340 steel, with material properties of Young's modulus $E = 3.0 \times 10^7$ psi and Poisson's ratio $\nu = 0.3$. Note that the FEM's coordinate systems were identical to the body reference frame of the roadarm in the dynamic model. Therefore, the loading history generated from the dynamic analysis could be used without transformation.

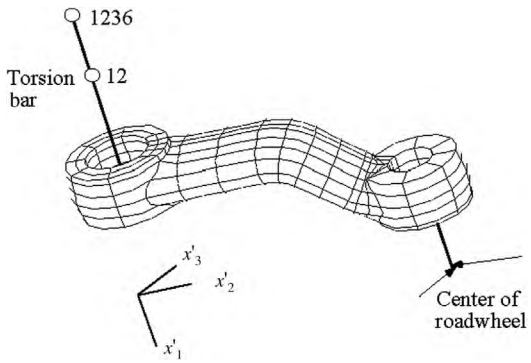


FIGURE 9.29

Roadarm finite element model.

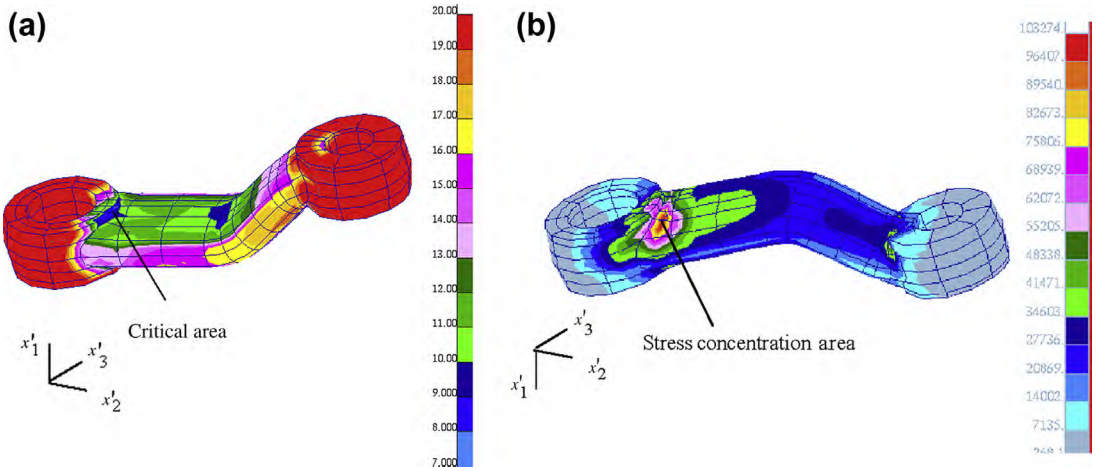


FIGURE 9.30

Analysis results: (a) fringe plot of crack initiation life and (b) fringe plot of static von Mises stress at 17.35 sec.

Finite element analysis was first performed to obtain stress influence coefficients for the roadarm using ANSYS. Eighteen quasistatic loads were applied, as discussed in Section 9.6. Among the loads, the first six that corresponded to external joint forces were three unit forces and three unit torques applied at the center of the roadwheel, in the x_1'-, x_2'-, and x_3'-directions, and the remaining 12 that corresponded to inertia forces were unit accelerations, unit angular accelerations, and unit combinations of angular velocities, as also discussed in Section 9.6. The stress influence coefficients obtained from analyses were six component stresses at finite element nodes in the x_1', x_2', and x_3' coordinates. Dynamic stresses at finite element nodes were then calculated by superposing stress influence coefficients with their corresponding external forces and accelerations and velocities in the time domain. To compute the multiaxial crack initiation life of the roadarm, the equivalent von Mises stress approach discussed in Section 9.3.3 was employed. The fatigue life contour is given in Figure 9.30(a). Note that the spectrum in Figure 9.30(a) is the number of blocks to initiate a crack in logarithm. A static stress fringe plot shown in Figure 9.30(b) demonstrates that the worst-case scenario employed for durability design using stresses is problematic. The stress fringe plot shown in Figure 9.30(b) was obtained by applying the peak load found at 17.35 sec of the 20-sec simulation, including six joint reaction forces at the roadwheel end, and accelerations, angular accelerations, and angular velocities of the roadarm. Note that, from Figures 9.30(a) and (b), the stress concentration area identified as the worst case did not conform to the critical areas where the crack was first initiated. Design based on maximum stress might not address the more critical issue of fatigue.

9.8.2 CASE STUDY: ENGINE CONNECTING ROD

Consider an engine connecting rod (Section 9.5.5). The thickness was assumed to be 1 mm. Material properties for the rod were as follows: Young's modulus $E = 210$ GPa, yield strength $S_y = 210$ MPa,

Poisson's ratio $\nu = 0.3$. The Paris constants were $C = 5.6 \times 10^{-12}$ mm/cycle and $m = 3.5$, and the fracture toughness was $K_{Ic} = 100$ MPa$\sqrt{\text{mm}}$. The load acting on the connecting rod in terms of the rotation angle θ is given by Eq. 9.68 (Hwang et al., 1997).

$$T_F = \begin{cases} 43.794\,\theta^2 + 30.19 \text{ at left inner circle } \left(\dfrac{-40}{180}\pi \le \theta \le \dfrac{40}{180}\pi\right) \\ 9.54\,\theta^2 - 42.97 \text{ at right inner circle } \left(\dfrac{140}{180}\pi \le \theta \le \dfrac{220}{180}\pi\right) \end{cases} \tag{9.68}$$

Figure 7.14(a) shows the FEM model of the rod with 6,058 DOF, which is considered adequate for stress analysis in general; in particular, the mesh was refined at the high-stress areas. The average element length ℓ_e in the crack region for this mesh was about 1.9 mm. This, of course, varied from element to element and is only given here to provide a rough idea of element size. The maximum principal stress distribution in the connecting rod is shown in Figure 7.14(b). Although the maximum stress appears to be at the fixed node on the left side, it is merely an artificial stress concentration due to displacement constraints. The real maximum stress of $\sigma_{\max} = 124$ MPa occurs on the left semicircular edge of the slot. An initial crack of $a = 7$ mm was introduced in this location and crack propagation analysis was conducted with $\Delta a = 1.5$ mm. Figure 9.31 shows the crack propagation path until failure—that is, until K_{eq} exceeded the fracture toughness of the material (Edke and Chang, 2010).

For mixed-mode cases, generally the crack propagation path is curvilinear. In this case, however, the path appeared to be zigzag due to alternating positive and negative signs of θ_c. As discussed in Section 9.5.4, the crack growth increment had to be reduced to minimize these oscillations.

From an analysis standpoint, XFEM-LSM does allow a crack to propagate within an element. However it is not advisable to use a Δa value that is quite small compared to element size. Thus, Δa and mesh size are interrelated and do have an impact on the accuracy of analysis results. This relationship also plays an important role in downstream design studies. If a large value Δa is selected, it may not accurately predict crack path and service life, leading to erroneous design. On the other hand, a very small value of Δa requires correspondingly fine mesh, thereby greatly increasing the computational burden.

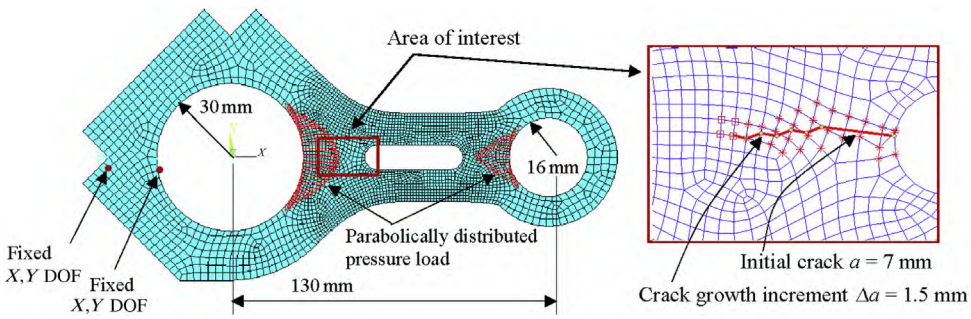


FIGURE 9.31

Finite element model of the connecting rod.

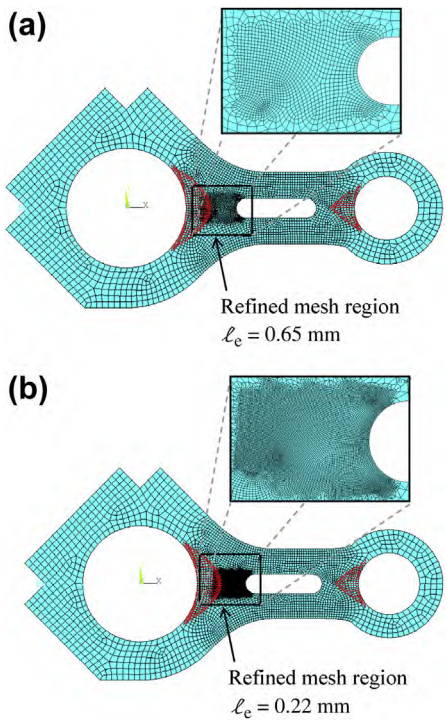


FIGURE 9.32

Finite element model of the connecting rod, (a) mesh 2: intermediate mesh, and (b) mesh 3: fine mesh.

For these reasons, a detailed study was undertaken to examine the effect of mesh refinement and crack growth increments on SIFs and the crack propagation path. Three different meshes (see Table 9.1) and 12 different crack growth increment sizes ($\Delta a = 0.1 - 1.2$ mm) were used. Mesh 1, the original mesh, is shown in Figure 9.31 and mesh 2 and mesh 3 are shown in Figures 9.32(a) and (b), respectively. Since the crack propagation path was roughly known, only the region near the crack propagation path was selected for refinement.

Because of the relation between mesh size and Δa size, not all values of Δa could be used for all meshes. In these cases, crack propagation analysis was conducted. Figures 9.33a–c show the crack

Table 9.1 Meshes and Corresponding Values of Δa for Connecting Rod

Variable	Degrees of Freedom	Average Element Length Near Crack (mm)	Range of Δa Values (mm)
Mesh 1	6058	1.9	0.9−1.2
Mesh 2	8554	0.65	0.3−1.2
Mesh 3	25,186	0.22	0.1−1.2

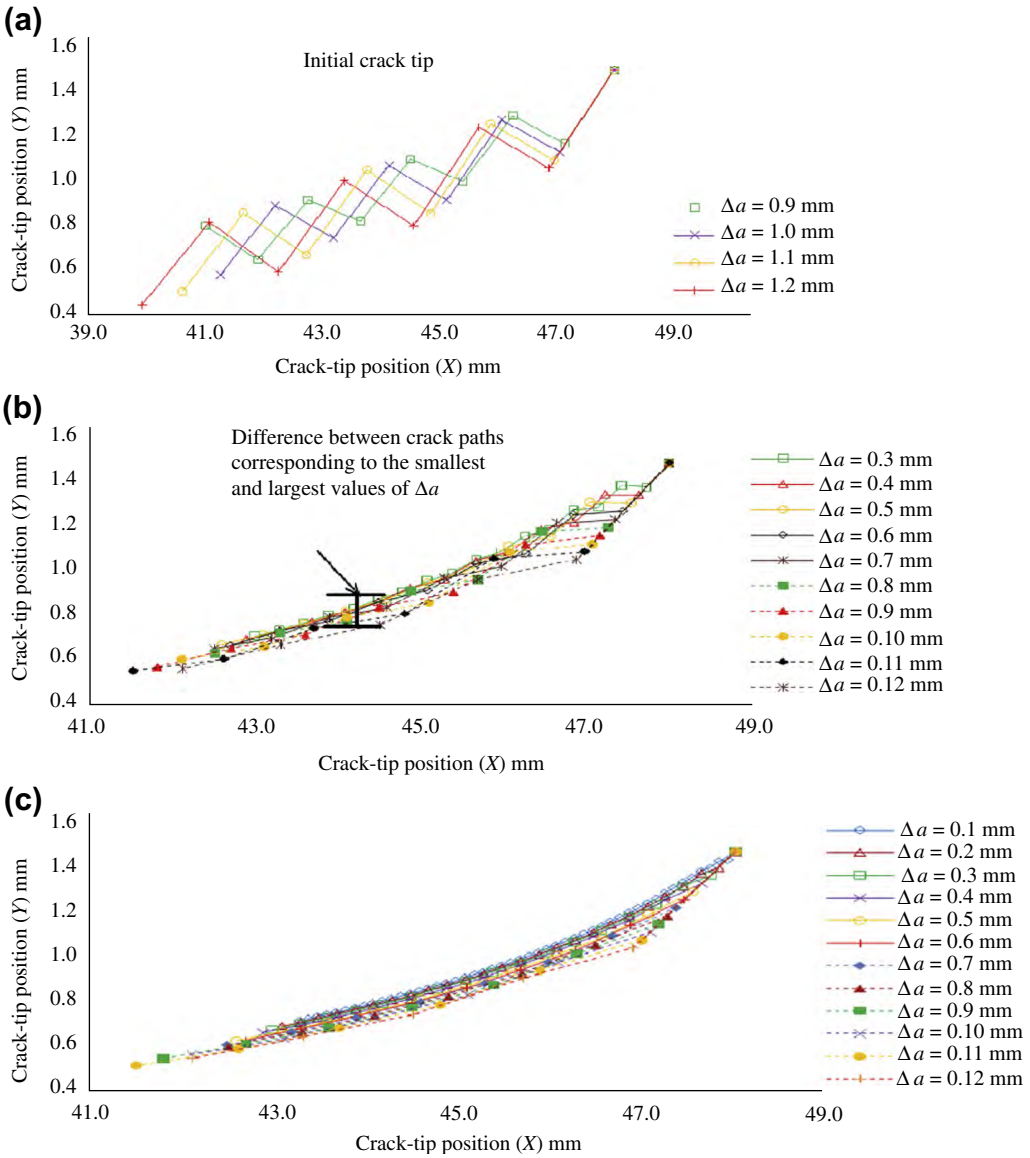


FIGURE 9.33

Crack propagation path for different values of Δa: (a) mesh 1, (b) mesh 2, and (c) mesh 3.

propagation path corresponding to different Δa values for the three meshes. The scale on both axes has been modified to clearly illustrate the effect. It can be observed that for mesh 1 decreasing Δa value did not have any noticeable impact on the oscillations. However, oscillations were significantly reduced

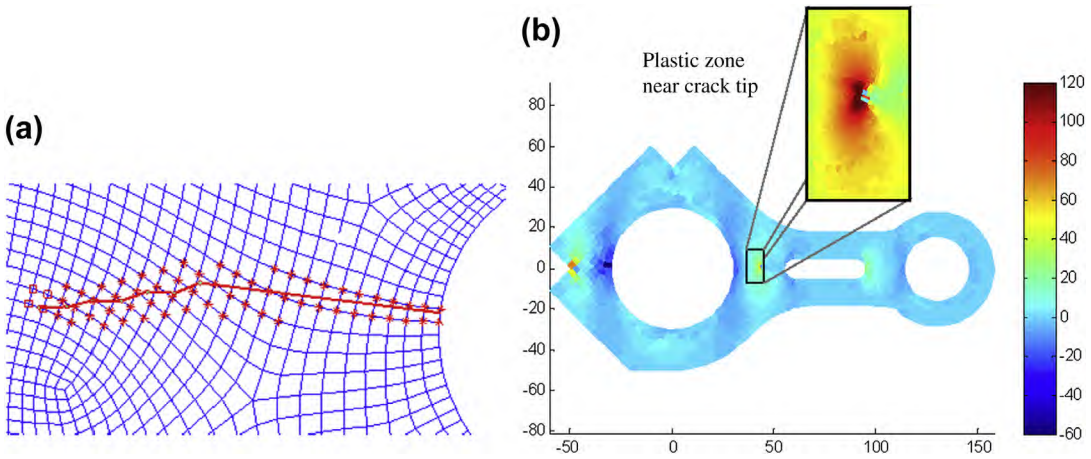


FIGURE 9.34

Analysis results of mesh 2 with $\Delta a = 0.8$ mm, (a) crack propagation path, and (b) *Y*-stress distributions (MPa).

from mesh 1 to mesh 2, especially for smaller values of Δa. For mesh 3, there were virtually no oscillations, and this was true for all values of Δa. Thus, the oscillations seemed to be more sensitive to mesh than Δa size.

Mesh 1 exhibited significant oscillation and hence was not suitable for accurate prediction of crack path. Mesh 2 did exhibit some oscillations, especially in the first few crack growth cycles, but the crack path predicted was very close to the oscillation-free crack path predicted by mesh 3, the finest mesh. It can thus be concluded that the additional computational burden associated with mesh 3 is not justified considering this minor difference in crack paths and that mesh 2 can be used for further studies.

While all Δa combinations for mesh 2 and mesh 3 were very close to each other, the curve for mesh 2 and the $\Delta a = 0.8$ mm case matched most closely with the best case. Hence, the combination of mesh 2 and $\Delta a = 0.8$ mm is adequate for further studies.

Fatigue lives of these three meshes with different Δa were also compared. As shown in Table 9.2, residual life estimated using different meshes and crack size increment Δa varied significantly. With the same mesh, a smaller Δa led to a less residual life, which tended to lead to more conservative designs. For a given Δa value, refined mesh again provided less residual life. The residual life of mesh 1 with $\Delta a = 1.2$ mm was almost twice that of mesh 3 with $\Delta a = 0.1$ mm, which was considered significant. In general, a safety factor of less than 2 is employed for mechanical components, such as those in ground vehicle suspensions. It is important to note that the large variation in residual life found in this example may not be adequately addressed in design using a safety factor approach.

For the selected mesh (mesh 2) and Δa size (0.8 mm), the crack propagation path and stress distribution at failure are shown in Figures 9.34(a) and (b), respectively. The crack growth path was quite smooth, as seen in Figure 9.34. The peculiar plastic zone shape for the plane strain condition

could be observed near the crack-tip region, as shown in Figure 9.34(b). The plastic zone size was quite small compared to the overall size of the connecting rod and hence the LEFM assumption was valid for this example.

9.8.3 TUTORIAL EXAMPLE: CRANKSHAFT

A crankshaft, which is a load-bearing component in a slider-crank mechanism and is shown in Figures 9.35(a), is employed as a tutorial example for HCF calculation using SolidWorks Simulation. The major geometric dimensions of the crankshaft are shown in Figure 9.35(b). Note that two small fillets (radius 0.05 in.) were added to the intersections between the two cylinders at the end and the crank body. A bearing load of 250 lb was added to the outer cylindrical surface of the

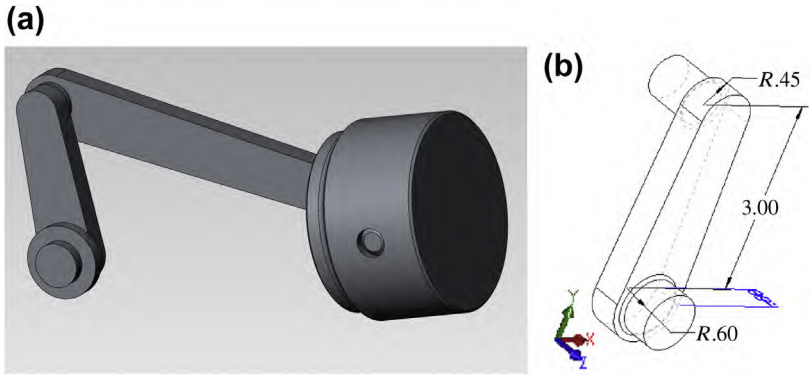


FIGURE 9.35

(a) Slider-crank mechanism and (b) crankshaft.

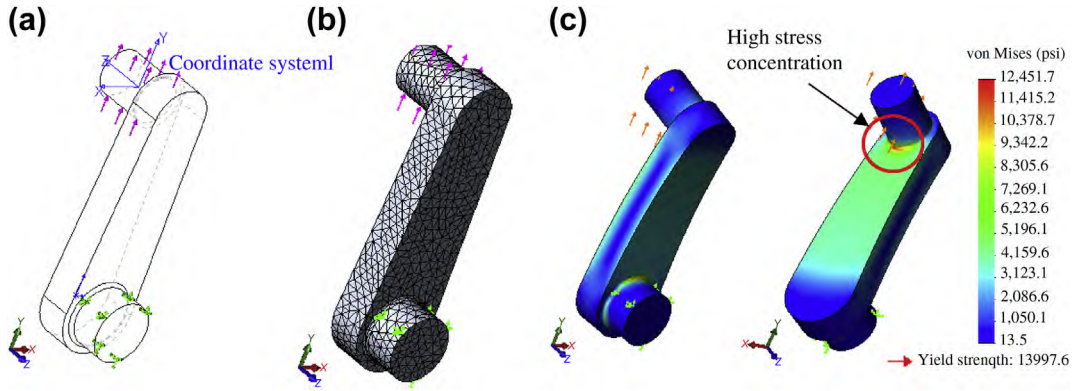


FIGURE 9.36

Crankshaft finite element model: (a) load and boundary condition, (b) finite element mesh, and (c) von Mises stress fringe plot.

Table 9.2 Residual Life Results for Different Meshes and Corresponding Values of Δa

Variable	Degrees of Freedom	Average Element Length Near Crack (mm)	Δa Value (mm)	Residual Life (cycles)
Mesh 1	6058	1.9	0.9	192,425
			1.2	213,170
Mesh 2	8554	0.65	0.3	124,137
			0.7	141,916
			0.8	143,640
			1.2	154,408
Mesh 3	25,186	0.22	0.1	116,650
			0.4	127,079
			0.8	139,643
			1.2	152,702

shaft (the cylinder on the top end), and the outer cylindrical surface of the lower cylinder was fixed, as shown in Figure 9.36(a). The crankshaft was meshed with 11,867 tetrahedral finite elements (Figure 9.36(b)). A static analysis was carried out, and the von Mises stress plot, shown in Figure 9.36(c), had a stress concentration with a stress level of 12,450 psi. Note that this stress was lower than the material yield strength of AL2014, $S_y = 13{,}998$ psi, provided in the SolidWorks material library.

An HCF analysis was carried out, assuming that the 250 lb bearing load was fully reversed. The equivalent von Moses stress method was chosen for fatigue calculation. For AL2014, the *S-N* diagram provided by SolidWorks Simulation is shown in Figure 9.37(a). The crack initiation fatigue life fringe plot is shown in Figure 9.37(b), in which the lowest life is located at the area where the maximum

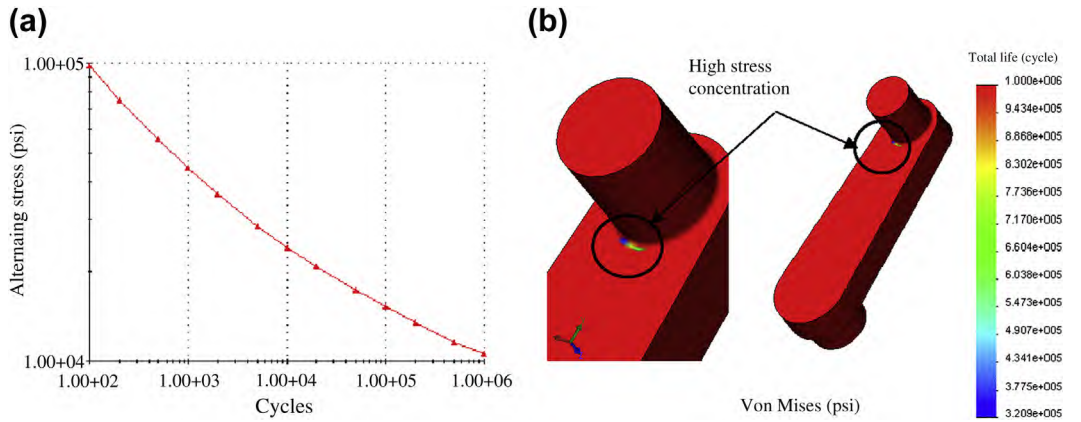


FIGURE 9.37

Fatigue life of the crankshaft: (a) *S-N* diagram and (b) fatigue life fringe plot.

von Mises stress occurs, as often expected. The lowest life is about 320,000 cycles. This result can be verified by drawing a horizontal line for the maximum stress 12,450 psi on the *S-N* diagram, intersecting the stress line with the *S-N* curve, and projecting the corresponding fatigue life on the horizontal axis.

9.9 SUMMARY

In this chapter, we discussed methods for fatigue analysis, both crack initiation and crack propagation. For crack initiation, there are stress-based and strain-based approaches for high- and low-cycle fatigue analyses, respectively. The stress-based approach is simple and easy to calculate, but it offers only a very rough estimate of fatigue life since it uses elastic stress as the input while physical fatigue initiation is driven by local plastic strains. Therefore, the stress-based approach is applicable to components with minimum and limited local plasticity areas. The strain-based approach considers local plastic strain and is more suitable for fatigue life calculations, especially when local strains are significant in structural components. We have to examine carefully the stress and strain of the structural components and choose an adequate method that provides reliable results for design decision making.

We also discussed fracture mechanics for crack propagation and fracture analysis. This discussion included the powerful *J*-integral for stress intensity factor calculation as well as mixed modes for 2D applications. In addition, we briefly covered a recently developed method that supports crack propagation analysis, XFEM. This method alleviates the need for remeshing when crack propagates, which is a huge advantage in tackling problems with complex geometry. The XFEM capability in ABAQUS offers superior capabilities for crack propagation computation.

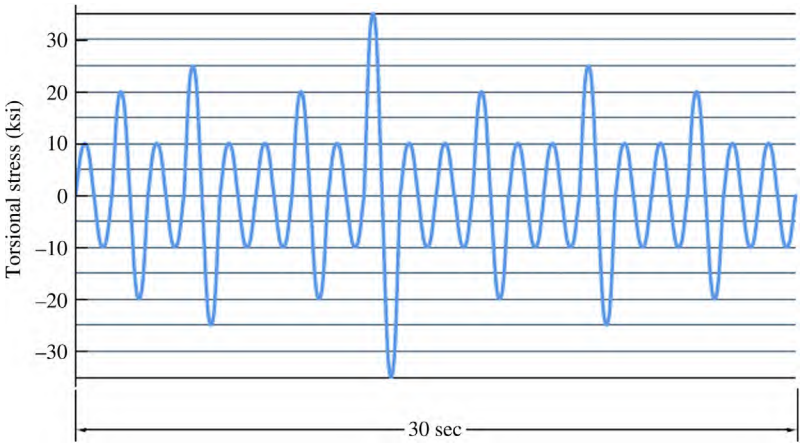
We briefly reviewed software tools that offer fatigue calculation capabilities, and provided case studies that give a general idea of the kind of applications that are possible in simulating fatigue life computation in general. We hope this chapter provided enough information to increase the reader's familiarity with fatigue analysis to address structural durability in engineering design. With more practice will come more confidence and competence in the use of software tools for carrying out crack initiation and crack propagation computations.

QUESTIONS AND EXERCISES

9.1. In Section 9.1 we discussed several famous incidents that involved fatigue failures. Please find and review three more incidents that involved failure caused by structural fatigue. Describe the incidents, identify the nature of the failures, and offer opinions in terms of preventing similar incidents.

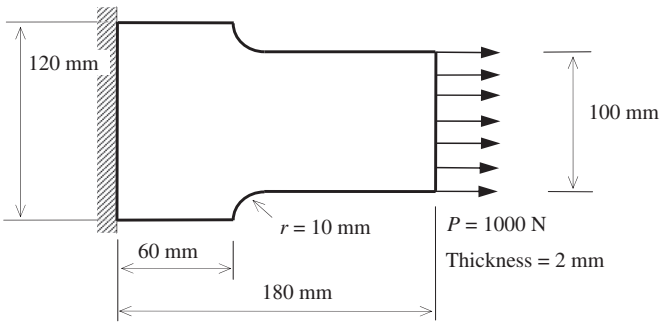
9.2. A 20-mm-diameter shaft transmits a variable torque of ± 400 N-m. The frequency of the torque variation is 0.1 sec^{-1}. The shaft is made of high-carbon steel (AISI 1080, $S_{ut} = 615$ MPa). Find the fatigue life of the shaft (in hours).

9.3. For the shaft in Exercise 9.2, we assume that the loading is one of completely reversed torsion. During a typical 30 sec of operation under overload conditions, the nominal stress was calculated to be as shown in the figure below. Estimate the life of the shaft when it is operating continually under these conditions.



9.4. A flood-protection dam gate is supposed to operate only once per week for 120 years, but after 40 years of use it needs to be operated twice per day (each time the high tide comes in). Determine how much lower the bending stress must be from then on to still give a total life of 120 years. The material being fatigued is medium-carbon steel (AISI 1040, $S_{ut} =$ 520 MPa).

9.5. A plate with two fillets is loaded with a cyclic tensile load $P = 1000$ N, as shown in the following figure. The material properties of the bar that are relevant to the calculation have been found to be $K = 155{,}000$ psi, $n = 0.15$, $\varepsilon_f' = 0,48$, $\sigma_f' = 290,000$ psi, $a = -0.091$, and $\alpha = -0.60$. How many cycles would it take to initiate a fatigue crack at the fillet root?



REFERENCES

Anderson, T.L., 1994. Fracture Mechanics: Fundamentals and Applications, second ed. CRC Press.
Bannantine, J.A., Comer, J.J., Handrock, J.L., 1990. Fundamentals of Metal Fatigue Analysis. Prentice Hall.

Bannerman, D.B., Young, R.T., 1946. Some improvements resulting from studies of welded ship failures. Welding Journal 25 (3), 223–236.

Belytschko, T., Black, T., 1999. Elastic crack growth in finite elements with minimal remeshing. International Journal for Numerical Methods in Engineering 45 (5), 601–620.

Brown, M.W., Miller, K.J., 1973. A theory for fatigue failure under multiaxial stress-strain conditions. Proceedings, Institution of Mechanical Engineering 187 (1), 745–755.

Buch, A., May–June 1997. Prediction of the comparative fatigue performance for realistic loading distributions. Progress in Aerospace Sciences 33 (5–6), 391–430.

Chang, K.H., Yu, X., Choi, K.K., 1997. Shape Design Sensitivity Analysis and Optimization For Structural Durability. International Journal of Numerical Methods in Engineering 40, 1719–1743.

Chu, C.C., Conle, F.A., Bonnen, J.J.F., 1993. Multiaxial stress-strain modeling and fatigue life prediction of sae axle shafts. In: McDowell, E. (Ed.), Advances in Multiaxial Fatigue. ASTM STP 1191. ASTM, Philadelphia, PA, pp. 37–54.

Coffin, L.F., 1954. A study of the effects of cyclic thermal stresses on a ductile metal. Transaction of ASME 76, 931–950.

Collins, J.A., 1993. Failure of Materials in Mechanical Design: Analysis, Prediction, Prevention, second ed. John Wiley.

Dang Van, K., Griveau, B., Message, O., 1989. On a new multiaxial fatigue limit criterion: theory and application, biaxial and multiaxial Fatigue. Mechanical Engineering Publications, London. EGF 3479–3496.

Dang Van, K., Papadopoulos, Y.V., 1987. Multiaxial fatigue failure criterion: a new approach. In: Proceedings of the Third International Conference on Fatigue and Fatigue Thresholds, Fatigue, 87. University of Virginia, Charlottesville.

Dowling, N.E., 2007. Mechanical Behavior of Materials, Engineering Methods for Deformation, Fracture and Fatigue, third ed. Pearson Education.

Downing, S.D., Socie, D.F., 1982. Simple rainflow counting algorithm. International Journal of Fatigue 4 (1), 31–40.

Edke, M., Chang, K.H., 2010. Shape Sensitivity Analysis For 2-D Mixed Mode Fractures Using Extended FEM (XFEM) And Level Set Method (LSM), Mechanics Based Design of Structures and Machines. submitted, accepted for publications in June 2009 38(03), 328–347. 2010.

Edke, M., Chang, K.H., 2011. Shape Optimization For 2-D Mixed Mode Fractures Using Extended FEM (XFEM) And Level Set Method (LSM). Structural and Multidisciplinary Optimization 44 (2), 165–181. http://dx.doi.org/10.1007/s00158-010-0616-5.

Endo, T., Matsuiski, M., March 1968. Fatigue of metals subjected to varying stress. Paper presented at the Kyushu District Meeting of the Japan Society of Mechanical Engineers, No. 68-2. Fukuoka Japan, pp. 37–40, in Japanese.

Fatemi, A., Socie, D., 1988. A critical plane approach to multiaxial fatigue damage including out-of-phase loading. Fatigue and Fracture of Engineering Materials and Structures 11 (3), 149–165.

Fjeldstad, A., Wormsen, G., 2006. Härkegård Simulation of fatigue crack growth in components with random defects.

Forman, R.G., Hearney, V.E., Engle, R.M., 1967. Numerical analysis of crack propagation in cyclic-loaded structures. Journal of Basic Engineering Trans of ASME 89, 459–464.

Forman, R.G., Mettu, S.R., 1992. Behavior of surface and corner cracks subjected to tensile and bending loads in TI-6AL-4 Valloy, Fracture Mechanics 22nd Symposium. In: Ernst, H.A., Saxena, A., McDowell, D.L. (Eds.), American Society for Testing, Materials.

Griffith, A.A., 1921. The phenomena of rupture and flow in solids. Philosophical Transactions of the Royal Society of London A 221, 163–198.

Halfpenny, A., 2005. A practical discussion on fatigue. HBM nCode, www.ncode.com.

Harter, J.A., 1994. MODGRO Version 1.2, Technical Memorandum AFWALTM-88-157-FIBE. Wright-Patterson AFB, AFWAL Flight Dynamics Laboratory.

Hertzberg, R.W., 1983. Deformation and fracture mechanics of engineering materials. Wiley.

Hoffmann, M., Seeger, T., October 1985. A Generalized Method for Estimating Multiaxial Elastic-Plastic Notch Stresses and Strains, Part 1: Theory. Journal of Engineering Materials and Technology 107 (4), 250–254.

Hwang, H.-Y., Choi, K.K., Chang, K.H., 1997. Shape Design Sensitivity Analysis and Optimization Using p-Version Finite Element Analysis. Mechanics of Structures and Machines 25 (No. 1), 103–137.

Irwin, G., 1957. Analysis of stresses and strains near the end of a crack traversing a plate. Journal of Applied Mechanics 24, 361–364.

Juvinall, R.C., Marshek, K.M., 2005. Fundamentals of Machine Component Design, fourth ed. Wiley.

Manson, S.S., 1953. Behaviour of Materials Under Conditions of Thermal Stress. NACA TN-2933.

Matake, T., 1977. An explanation on fatigue limit under combined stress. Bulletin JSME 20, 257–263.

Morrow, J., July 1965. Cyclic plastic strain energy and fatigue of metals. Internal Friction Damping and Cyclic Plasticity 45–84. ASTMSTP378.

National Bureau of Standards, 1983. The Economic Effects of Fracture in the United States. Department of Commerce, U.S.

National Transportation Safety Board, 1989. Aircraft Accident Report: Aloha Airlines, Flight 243, Boeing 737–200. N73711 near Maui, Hawaii. 28 April 1988.

National Transportation Safety Board, 1990. United Airlines Flight 232 McDonnell Douglas DC-10-10 Sioux Gateway Airport. Iowa, Sioux City.

Neuber, H., 1961. Theory of stress concentration for shear-strained prismatical bodies with arbitrary nonlinear stress-strain laws. Journal of Applied Mechanics E28, 544.

Paris, P.C., Gomez, M.P., Anderson, W.E., 1961. A rational analytic theory of fatigue. The Trend in Engineering 13, 9–14.

Ramberg, W., Osgood, W.R., 1943. Description of stress-strain curves by three parameters. National Advisory Committee for Aeronautics. Technical Note No. 902.

Rice, J.R., 1968. A path independent integral and the approximate analysis of strain concentration by notches and cracks. Journal of Applied Mechanics 35, 379–386.

Sanders, J.R., Tesar, D., 1978. The analytical and experimental evaluation of vibration oscillations in realistically proportioned mechanisms. ASME Paper No. 78-DE-1.

Shigley, J.E., Mischke, C.R., Budynas, R.G., 2004. Mechanical Engineering Design, seventh ed. McGraw-Hill.

Smith, J., Watson, P., Topper, T., 1970. A stress strain function for fatigue of metals. Journal of Materials 4 (5), 293–298.

Socie, D., 1993. Critical plane approaches for multiaxial fatigue damage assessment. In: McDowell, D.L., Ellis, R. (Eds.), Advances in Multiaxial Fatigue. ASTM International, pp. 7–36.

Spotts, M.F., Shoup, T.E., 1998. Design of Machine Elements, seventh ed. Prentice Hall.

Walker, K., 1970. The effect of stress ratio during crack propagation and fatigue for 2024-T3 and 7075-T6 aluminum. Paper ASTM STP 462. Proceedings of the American Society for Testing and Materials.

Yau, J.F., Wang, S.S., Corten, H.T., 1980. A mixed-mode crack analysis of isotropic solids using conservation laws of elasticity. Journal of Applied Mechanics 47, 335–341.

You, B.R., Lee, S.B., 1996. A critical review on multiaxial fatigue assessments of metals. International Journal of Fatigue 18 (4), 235–244.

CHAPTER

RELIABILITY ANALYSIS 10

CHAPTER OUTLINE

10.1 Introduction 525
10.2 Probability of Failure—Basic Concepts 526
10.2.1 Deterministic Design versus Probabilistic Prediction 527
10.2.2 Probabilistic Design 531
10.2.3 Short Summary 534
10.3 Basics of Statistics and Probabilistic Theory 535
10.3.1 Events and Basic Probability Rules 535
10.3.2 Random Variables and Distribution Functions 538
10.3.2.1 Random Variables 538
10.3.2.2 Distribution Functions 538
10.3.2.3 Mean Value and Standard Deviation 539
10.3.2.4 Joint Probability Density Function 539
10.3.3 Probabilistic Distributions 543
10.3.3.1 Normal Distribution 544
10.3.3.2 Lognormal Distribution 545
10.3.3.3 Extreme Value Distributions 546
10.4 Reliability Analysis Methods 547
10.4.1 The Limit State Function 548
10.4.2 Monte Carlo Simulation 549
10.4.3 The First-Order Reliability Method 552
10.4.3.1 FORM 553
10.4.3.2 The Reliability Index Approach 560
10.4.3.3 The Performance Measure Approach 563
10.4.4 The Second-Order Reliability Method 566
10.4.5 Transformation of Random Variables* 568
10.4.5.1 Correlated Random Variables of Normal Distribution 568
10.4.5.2 Independent Random Variables of Non-Normal Distribution 571
10.4.5.3 Correlated Random Variables of Non-Normal Distribution 573
10.4.6 Importance Sampling 575
10.4.7 The Response Surface Method 579
10.4.8 Short Summary 581

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00010-7

Copyright © 2015 Elsevier Inc. All rights reserved.

10.5 Multiple Failure Modes* **581**
10.5.1 Series System 582
10.5.2 Parallel System 583
10.5.3 FORM Approximation for a Series System 585
10.6 General-Purpose Reliability Analysis Tools **588**
10.7 Case Study **589**
10.8 Summary **591**
Questions and Exercises **593**
References **594**

We live in a world of uncertainties. When we roll a die, the outcome is uncertain. We know only that there is a 50% probability we will get an even (or odd) number. When we walk to work or school, we are often uncertain if it is going to rain later in the day. Meteorologists can tell us only that there is a 30% chance of rain, and they will never be wrong giving us such a probabilistic prediction.

In engineering design, there are uncertainties. Take a simple cantilever beam as an example. Uncertainties or variabilities exist in loading, material properties, geometric size, and material strength. However, when we calculate the maximum bending stress, we usually assume that all of the numbers we insert into the bending stress equation are deterministic; that is, there is no uncertainty in these quantities. When we are given a safety factor to determine if the design is safe, we compare the bending stress of the cantilever beam with its material strength to see if the ratio of strength to stress is greater than the given safety factor. We are usually satisfied with the design if the strength-to-stress ratio is greater than the required safety factor. However, is the design verified by the safety factor approach truly safe? Is the design verified truly reliable? Does a design with a safety factor greater than 1 never fail? Not likely? What if we were to increase the safety factor to 2 or greater? How safe is the design in reality? How reliable can a design be? How do we tell?

These questions can be answered only through a probabilistic approach. A deterministic approach using a safety factor or a worst-case scenario is not sufficient to address the safety or reliability of a product design. A safe and reliable product can be ensured only by considering probabilities or statistics in the design process.

Mechanical engineers must understand the importance of the probabilistic aspect of product design, and must be able to apply adequate reliability analysis methods to engineering problems. This chapter is devoted to the subject of reliability analysis. We also touch on design from a probabilistic perspective, although a more in-depth discussion of reliability-based design is provided in Chapter 19 Multiobjective Optimization and Advanced Topics.

Although reliability analysis is the focus of this chapter, the topic is no doubt a substantial one—usually an entire book is necessary for a comprehensive treatment. Because providing all of the details of reliability analysis in just one chapter is not feasible, this chapter is organized to focus on introducing the most popular and powerful of its methods. Therefore, the first-order reliability analysis method (FORM), the second-order reliability method (SORM), Monte Carlo simulation, importance sampling, and the response surface method are introduced with sufficient detail to help the reader understand them and apply them to engineering applications. See Choi et al. (2006) for comprehensive discussion of the subject.

One important assumption made in this chapter is that readers are familiar with the basics of engineering statistics, such as random variables, the probability density function, the cumulative distribution function, and so forth. These are briefly reviewed in Section 10.3. Readers are encouraged to review this section before attempting to grasp concepts and mathematic formulations for the reliability analysis methods introduced in the subsequent sections.

Another point worth mentioning is that we assume that all uncertainties are irreducible. In general, uncertainties may be classified as reducible or irreducible. Reducible uncertainties are normally caused by lack of data, modeling simplifications, human errors, and the like, and can usually be handled by, among other things, collecting more data, improving analysis models with a better understanding of the problem, and implementing stricter quality control. Irreducible uncertainties, on the other hand, are caused by phenomena related to the stochastic nature of the physical problem and cannot be reduced by more knowledge or data. Addressing reducible uncertainties for reliability analysis is beyond the scope of this chapter. We discuss only irreducible uncertainties.

In addition to reliability analysis methods, this chapter discusses general-purpose software tools for carrying them out. A practical example, a tracked vehicle roadarm, is included as a case study to illustrate and demonstrate the reliability analysis methods discussed. Almost all examples included in this chapter are structural problems to maintain focus. However, theory and methods introduced are applicable to other types of engineering problems.

Overall the objectives of this chapter are as follows, to provide basic probabilistic theory and reliability analysis methods using simple examples; to explain the importance of the probabilistic nature of engineering design; to enable the reader to apply the analysis methods to basic engineering problems; to provide a basic knowledge of reliability analysis software.

10.1 INTRODUCTION

In engineering design, the traditional deterministic approach has been successfully applied to systematically reduce cost and improve product performance. However, the existing uncertainties or variabilities in physical quantities, such as loads, materials, manufacturing tolerance, and so forth, are unavoidable in engineering design. These uncertainties must be considered in the product design process.

The traditional way of dealing with uncertainties is to use conservative values of the uncertain quantities as well as safety factors in the framework of deterministic design. Conservative values lead to a product that is often overdesigned and therefore heavy and inefficient. A safety factor approach offers a safety measure that is only relative. Although the use of a safety factor is satisfactory with most design applications, determining an adequate safety factor for a given design problem is uncertain itself. A larger safety factor usually makes the product "safer"; however, if unnecessarily large it too often yields an overdesigned product, which is not desirable. What level of safety factor is considered to be just right? How safe is the design obtained using a safety factor? This safety factor approach does not provide design engineers with a full picture of the reliability level of the product. The question we should ask is not if the design will fail but instead what the probability is that it will fail. This question can only be answered from a probabilistic perspective. Reliability analysis is key in addressing the safety and reliability in product design.

Reliability analysis deals primarily with the effects of random variability on the performance of an engineering component or system during the design phase. To carry out a reliability analysis, a failure mode must first be identified. A failure mode describes how the product or component fails; it indicates that response exceeds the component's or system's design limit. For example, when the maximum stress of a cantilever beam exceeds its material strength, the beam fails. In this case, the maximum stress performance of the beam is defined as the failure mode.

After a failure mode is defined, variabilities of the physical parameters or manufacturing process that affect stress and strength must be identified. For example, the length and cross-section dimensions of the beam can vary depending on the tolerance requirements in its manufacture. Material properties, such as elasticity modules or yield strength, are uncertain and depend on the manufacturing process of the raw material. Certainly, the load applied to the beam can vary according to how the beam is loaded and how the load is controlled. Reliability analysis takes these uncertainties into consideration and estimates a failure probability that predicts the percentage of incidents in which the maximum stress of the beam exceeds its strength. There is no need to determine a safety factor. The result offered by reliability analysis is far more precise and effective than that provided by a safety factor.

A number of methods have been developed to support reliability analysis in product design. In particular, for probabilistic structural design, Monte Carlo simulation, the first-order reliability method, the second-order reliability method, importance sampling, and the response surface method, among others, have been applied to solve practical applications on a broad basis. Our emphasis in this chapter is on structural problems. To stay focused we use mostly structural examples to illustrate concept, theory, and methods, which, however, are applicable to other engineering problems that involve different failure modes and associated performance measures.

In this chapter, we start by introducing the basic concept of probability of failure using simple examples. Here we compare deterministic and probabilistic approaches to demonstrate that reliability is indispensable in product design. A brief review of essential topics in engineering statistics relevant to our discussion is provided in Section 10.3. Readers are encouraged to review this section before moving to later sections, which are more involved with statistics theory and require a good prior understanding. The key part of this chapter is Section 10.4, in which reliability analysis methods are introduced. We also briefly discuss system reliability and present failure probability prediction for a series system using FORM. We discuss reliability analysis software in Section 10.6. Example problems modeled and solved using some of the methods discussed are presented. The chapter wraps up with a case study.

10.2 PROBABILITY OF FAILURE—BASIC CONCEPTS

We start our discussion with the basic concepts of failure probability. The goal is to introduce key ideas in reliability analysis using a simple cantilever beam, avoiding sophisticated math and theory at the beginning. We first recall the safety factor approach that we are familiar with in deterministic design, we will point out a few of its shortcomings. We then see how a probabilistic approach can be applied to the same beam example, in which the adequacy of the approach is demonstrated. We also briefly touch on probabilistic design by adjusting the beam dimension in an attempt to reduce its failure probability. We compare the results with those of the worst-case approach commonly employed in deterministic design.

10.2.1 DETERMINISTIC DESIGN VERSUS PROBABILISTIC PREDICTION

In engineering analysis and design using a deterministic approach, we assume the physical parameters to be deterministic. When we say the length of a cantilever beam is 10 in., we assume that every single beam manufactured is exactly 10 in. long. In fact, not a single beam is exactly 10 in. long; in fact, 10 in. is a nominal value we use. In statistics, this value often refers to an average or a mean of the length parameter, which can be obtained by measuring the length of a bulk of cantilever beams that were manufactured following the same design specifications. Similarly, material yield strength, which can be found in a strength of materials textbook, is a mean value. In deterministic design, we are essentially employing mean values of the physical parameters for our calculations.

For a steel cantilever beam of solid circular cross-section, shown in Figure 10.1, the length and diameter are given as 10 in. and 1 in., respectively. The load applied at the tip is 393 lb. With these values, the maximum bending stress can be calculated as

$$s = \frac{32M}{\pi d^3} = \frac{32(p\ell)}{\pi d^3} = \frac{32(393)(10)}{\pi(1)^3} = 40{,}000 \text{ psi} = 40 \text{ ksi} \tag{10.1}$$

If the yield strength of the steel is $S_y = 50$ ksi, the safety factor can be calculated as $n = S_y/s = 50/40 = 1.25$. This is simple and almost effortless. But the question is whether we are satisfied with the design. Usually we are, since the safety factor is greater than 1. As a result, we are under the impression that the beam will not fail; that is, the bending stress will not be greater than the yield strength.

Is this true? Are we sure? Is the design really safe? How safe? Is safety factor $n = 1.25$ good enough? Will we see any failure; in other words, is there any possibility that the bending stress will be greater than the yield strength if hundreds or thousands of the cantilever beams are manufactured and tested? If the beam fails, how often does failure occur? If we increase the safety factor from 1.25 to 1.5 by, for example, using a stronger material with a higher yield strength, how much improvement in terms of reducing the possibility of failure can we expect?

These are valid questions and the very ones we should ask as design engineers. We have to answer them by calculating probability of failure using reliability analysis. Mathematically, the probability of failure of this stress failure mode for the cantilever beam can be defined as

$$P_f = P(s > S_y) = P(S_y - s \leq 0) \tag{10.2}$$

where $P(\bullet)$ is the probability of the failure mode $\bullet$, and the failure mode in this case is defined as $S_y - s \leq 0$. Equation 10.2 is referred to as a simple strength-load (or strength-stress) approach for reliability analysis, and the equation $S_y - s = 0$ is referred to as the limit state function.

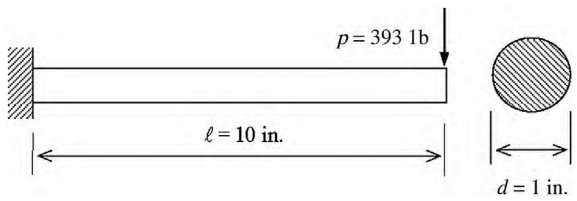


FIGURE 10.1

Cantilever beam of solid circular cross-section.

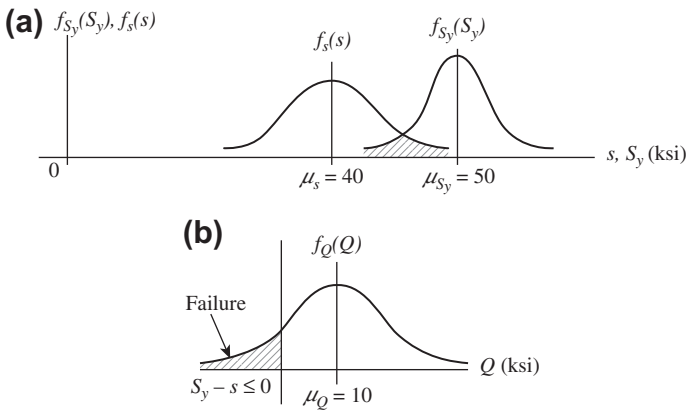


FIGURE 10.2

Probability distributions of (a) bending stress s and yield strength S_y and (b) failure mode $Q = S_y - s$.

Next, we assume both stress s and strength S_y to be random variables of normal distribution; that is, both have a distribution like the bell-shape curve shown in Figure 10.2. The stochastic characteristics of the stress and strength are usually obtained from experiments. We assume that the mean value and standard deviation for the yield strength are $\mu_{s_y} = 50$ ksi and $\sigma_{s_y} = 6$ ksi, respectively. Notice that the bending stress is affected by load and geometric dimensions. Variabilities in these parameters affect its stochastic characteristics. For the time being, we simply assume that the mean value and standard deviation of the bending stress are $\mu_s = 40$ ksi and $\sigma_s = 8$ ksi, respectively.

The distribution functions $f_{s_y}(S_y)$ and $f_s(s)$ shown in Figure 10.2(a) are the respective probability density functions (PDFs) of the random variables S_y and s. A probability density function describes the relative likelihood for this random variable to take on a given value. The area underneath each curve is 1, which represents the 100% probability of the entire sample space. The overlapped area underneath the two intersecting curves represents beam failure $S_y - s \le 0$; that is, bending stress s is greater than yield strength S_y, meaning that the overlapped area represents the probability of failure. The question is how we calculate the probability of failure.

We first rewrite the failure mode as

$$Q = S_y - s \tag{10.3}$$

where Q is a random variable with normal distribution since both S_y and s are assumed normally distributed. The probability density function $f_Q(Q)$ of the random variable Q is shown in Figure 10.2(b), in which the area underneath the curve to the left of the origin represents the beam failure. Therefore, the probability of failure P_f can be calculated as

$$P_f = \int_{-\infty}^{0} f_Q(Q)\,dQ \tag{10.4}$$

where $f_Q(Q)$ is the normally distributed PDF for Q and can be written mathematically as

$$f_Q(Q) = \frac{1}{\sigma_Q\sqrt{2\pi}}\, e^{-\frac{1}{2}\left(\frac{Q-\mu_Q}{\sigma_Q}\right)^2} \tag{10.5}$$

where μ_Q and σ_Q are the respective mean value and standard deviation of the random variable Q.

Instead of using Eq. 10.4 to calculate P_f directly, it is easier and more common to transform the PDF of a normal distribution $f_Q(Q)$ to a standard (or normalized) normal distribution $\phi(z)$, which has mean value $\mu_z = 0$ and standard deviation $\sigma_z = 1$, as depicted in Figure 10.3. The transformation can be carried out simply by

$$z = \frac{Q - \mu_Q}{\sigma_Q} \tag{10.6}$$

and the standard normal distribution function is

$$\phi(z) = \frac{1}{\sqrt{2\pi}}\, e^{-\frac{1}{2}z^2} \tag{10.7}$$

Thus, failure probability can be calculated as

$$P_f = \int_{-\infty}^{0} f_Q(Q)dQ = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}t^2} dt = \Phi(z) \tag{10.8}$$

where $\Phi(z)$ is the cumulative distribution function (CDF) of $\phi(z)$. Note that $\Phi(z)$ for a given z can be found in engineering statistics textbook. Likewise, the function value for $\Phi(z)$ can be found using any engineering software with statistics capabilities, such as MATLAB. The MATLAB function for the value of the normal cumulative distribution function is `normcdf(z)`. For example, `normcdf(0)` = 0.5.

Now let us go back to the cantilever beam. The mean value and standard deviation of the random variable Q can be calculated, respectively, as

$$\mu_Q = \mu_{S_y} - \mu_s = 50 - 40 = 10 \text{ ksi} \tag{10.9a}$$

$$\sigma_Q = \sqrt{{\sigma_{S_y}}^2 + {\sigma_s}^2} = \sqrt{6^2 + 8^2} = 10 \text{ ksi} \tag{10.9b}$$

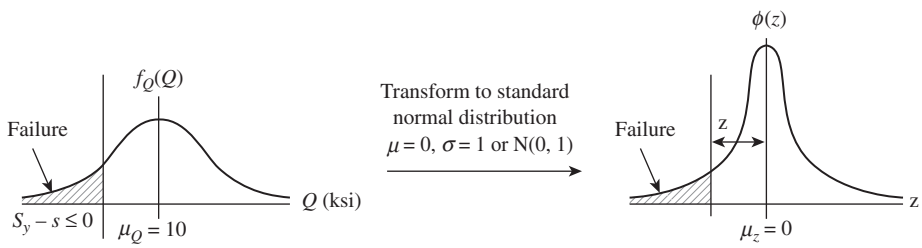


FIGURE 10.3

Transformation of a random variable of normal distribution into standard normal distribution.

Thus,

$$z = \frac{Q - \mu_Q}{\sigma_Q} = \frac{0 - 10}{10} = -1$$

and the probability of failure can be calculated using Eq. 10.8 as

$$P_f = \Phi(-1) = 0.159 = 15.9\%$$

The result shows that there is a 15.9% probability that the beam will fail, in which its maximum bending stress exceeds its yield strength. Note that Eqs 10.9a and 10.9b are simply the results of subtracting two random variables. For a combination other than subtraction, refer to Table 10.1. Derivations of these equations can be found in a statistics textbook.

It is apparent that the probability of failure calculated by reliability analysis can more precisely describe the reliability of the cantilever beam. The result shows that a safety factor $n = 1.25$ does not guarantee that the beam will never fail. A safety factor can only go so far in providing a relative measure in terms of the reliability of a design.

Now we know that the beam will fail with a quite high probability. The question is how we reduce the failure probability, for example, by using a stronger material. If we use a stronger steel with yield strength $S_y = 60$ ksi, the corresponding safety factor can be calculated as $n = 60/40 = 1.5$, increased from 1.25 if we keep the same beam dimensions. In this case, the failure probability can be calculated as follows. First the mean value of the random variable Q increases to 20 ksi as in the following equation.

$$\mu_Q = \mu_{s_y} - \mu_s = 60 - 40 = 20 \text{ ksi}$$

Table 10.1 Mean Value and Standard Deviation of Combined Independent Random Variables x and y

Functions	μ_Q	σ_Q
$Q = C$	C	0
$Q = cx$	$c\mu_x$	$c\sigma_x$
$Q = x + c$	$\mu_x + c$	σ_x
$Q = x \pm y$	$\mu_x + \mu_y$	$\sqrt{\sigma_x^2 + \sigma_y^2}$
$Q = xy$	$\mu_x\mu_y$	$\sqrt{\mu_x^2\sigma_y^2 + \mu_y^2\sigma_x^2}$
$Q = x/y$	μ_x/μ_y	$\dfrac{\sqrt{\mu_x^2\sigma_y^2 + \mu_y^2\sigma_x^2}}{\mu_y^2}$
$Q = \dfrac{1}{x}$	$\dfrac{1}{\mu_x}$	$\dfrac{\sigma_x}{\mu_x^2}$

Note that σ_Q stays the same since we are not changing any of the standard deviation of the stress and strength. Hence,

$$z = \frac{-20}{10} = -2$$

and

$$P_f = \Phi(-2) = 0.0228 = 2.3\%$$

That is, only 2.3% of the beams will fail, which is a significant improvement over the previous 15.9% failure rate. Is using stronger steel the only way to reduce failure probability? What if we tighten the tolerance in the manufacturing process? If we do so, let us assume that the standard deviation of the bending stress is reduced from 8 to 4 ksi (recall that the bending stress is a function of beam dimensions). We assume the same steel of yield strength $S_y = 40$ ksi as before, so $\mu_Q = 10$ ksi and

$$\sigma_Q = \sqrt{{\sigma_{S_y}}^2 + {\sigma_s}^2} = \sqrt{6^2 + 4^2} = 7.21 \text{ ksi} \tag{10.10}$$

Therefore, $z = \dfrac{-10}{7.21} = -1.39$, and the probability of failure is

$$P_f = \Phi(-1.39) = 0.0808 = 8.08\%$$

which is lower than the previous 15.9%.

Failure probability is reduced when manufacturing quality is improved by tightening tolerance. Note that this improvement has nothing to do with the safety factor (which is still $n = 1.25$) and has nothing to do with stronger material (yield strength is the same, $S_y = 40$ ksi).

There are several possible scenarios that can help us reduce the standard deviation of the bending stress from 8 to 4 ksi. To close out this discussion, we mention one possibility that involves manufacturing tolerance. We assume that beam length is a random variable of normal distribution with mean value and standard deviation $\mu_\ell = 10$ in. and $\sigma_\ell = 2$ in. respectively. We also assume that the load and diameter of the cross-section of the beam are deterministic. The bending stress is thus

$$s = \frac{32(p\ell)}{\pi d^3} = \frac{32(393)\ell}{\pi(1)^3} = 4000\ell \text{ psi} = 4\ell \text{ ksi}$$

If $\mu_\ell = 10$ in. and $\sigma_\ell = 2$ in. the mean value and standard deviation of the bending stress are $\mu_s = 4\mu_\ell = 40$ in. and $\sigma_s = 4\sigma_\ell = 8$ in., respectively. If we tighten the manufacturing tolerance that reduces the standard deviation of the beam length from 2 to 1 in. the standard deviation of the bending stress becomes $\sigma_s = 4\sigma_\ell = 4$ in. which gives $\sigma_Q = 7.21$ ksi as shown in Eq. 10.10.

What if two or more parameters in the bending stress equation are random? Their uncertainties affect the stochastic characteristics of the bending stress. How do we calculate failure probability of the beam when we have more random variables? This issue is formally addressed in Section 10.4, where we introduce reliability analysis methods.

10.2.2 PROBABILISTIC DESIGN

In this subsection we touch a bit on the subject of probabilistic design, in which effects of random variability on the performance of an engineering system or component are considered in product

design. Instead of offering a comprehensive discussion of this subject, we simply bring you some ideas about engineering design considering uncertainties. A more in-depth discussion, commonly referred to as reliability-based design, can be found in Chapter 19 Multiobjective Optimization and Advanced Topics.

Previously, we demonstrated that reducing the standard deviation of the beam length by tightening the manufacturing tolerance reduces failure probability. From the perspective of design, we often vary geometric dimensions to achieve better product performance. As mentioned before, usually the mean values of the respective dimensions are employed for deterministic design. In this subsection we use the same cantilever beam to illustrate the concept of probabilistic design.

We assume that the same random variable of yield strength: normal distribution with mean value and standard deviation $\mu_{S_y} = 50$ ksi and $\sigma_{S_y} = 6$ ksi, respectively. As before, we assume that the load and diameter of the beam cross-section are deterministic. Thus, the only random variable that affects the bending stress is the beam length, which has normal distribution with mean value and standard deviation μ_ℓ and σ_ℓ, respectively. We further assume that $\sigma_\ell = 0.1\mu_\ell$; as a result, we are left with one design variable, μ_ℓ.

Following the previous discussion we have

$$\mu_Q = \mu_{S_y} - \mu_s = 50 - 4\mu_\ell \tag{10.11a}$$

$$\sigma_Q = \sqrt{{\sigma_{S_y}}^2 + {\sigma_s}^2} = \sqrt{6^2 + (4\sigma_\ell)^2} = \sqrt{6^2 + (0.4\mu_\ell)^2} \tag{10.11b}$$

and

$$z = \frac{-(50 - 4\mu_\ell)}{\sqrt{6^2 + (0.4\mu_\ell)^2}} \tag{10.11c}$$

If the design requirement is to have a failure probability no greater than $P_f = 0.001$, the corresponding z value is

$$z = \Phi^{-1}(P_f) = \Phi^{-1}(0.001) = -3.09 \tag{10.12}$$

which again can be obtained from any statistics textbook or reference manual. If you use MATLAB, the function to call is `norminv(p)`, where the input parameter p is the failure probability P_f. For example, `norminv(0.001)` $= -3.09$.

If we equate Eqs 10.11c and 10.12, the mean value can be calculated as $\mu_\ell = 7.34$ in. If the failure probability is reduced ten times to $P_f = 0.0001$, $z = \Phi^{-1}(0.0001) = -3.72$. Then the mean value becomes $\mu_\ell = 6.43$ in. This result shows that a reduction of 0.91 in. (from 7.34 to 6.34) in beam length reduces the failure probability ten times, from $P_f = 0.001$ to $P_f = 0.0001$. We know that the safety factor and the failure probability are roughly related. As seen before, increasing the safety factor reduces failure probability. This relation can be more clearly illustrated for the cantilever beam as follows.

As mentioned earlier, mean values are employed for deterministic design. Therefore, the safety factor of the cantilever beam can be also written as

$$n = \frac{S_y}{s} = \frac{\mu_{Sy}}{\mu_s} = \frac{50}{4\mu_\ell} \tag{10.13}$$

Thus,

$$\mu_\ell = \frac{50}{4n} = \frac{12.5}{n} \tag{10.14a}$$

and

$$z = \frac{-50(1 - 1/n)}{\sqrt{6^2 + (5/n)^2}} \tag{10.14b}$$

$$P_f = \Phi(z) = \Phi\left(\frac{-50(1 - 1/n)}{\sqrt{6^2 + (5/n)^2}}\right) \tag{10.14c}$$

The mean value and failure probability can be graphed for a range of safety factors—for example, between 1 and 2, as shown in Figure 10.4, which clearly shows that increasing the safety factor reduces failure probability drastically. However, no matter how large the safety factor is, failure probability will never become zero. For example, when the safety factor increases to $n = 2$, the failure probability is 0.0059%, which is very small but not zero. It is important to keep in mind that reliability analysis offers much more precise information in failure probability estimates. The question is not if a safety factor $n = 2$ is sufficient; instead, the question is if a 0.005% failure rate is acceptable or if the rate is so small that the design can be relaxed a bit. It is evident that the probabilistic approach supports more informative design decision making than the deterministic approach in engineering design.

One last topic to discuss in this section is the absolute-worst-case method that we commonly employ for design following deterministic approach. We use the same cantilever beam to illustrate the approach and point out a few important points and shortcomings of the method.

With the absolute-worst-case method we assume that the tolerances of yield strength and diameter of the beam cross-section are, respectively,

$$S_y = \mu_{S_y} \pm 3\sigma_{Sy} = 50 \pm 3(6) \text{ ksi}$$

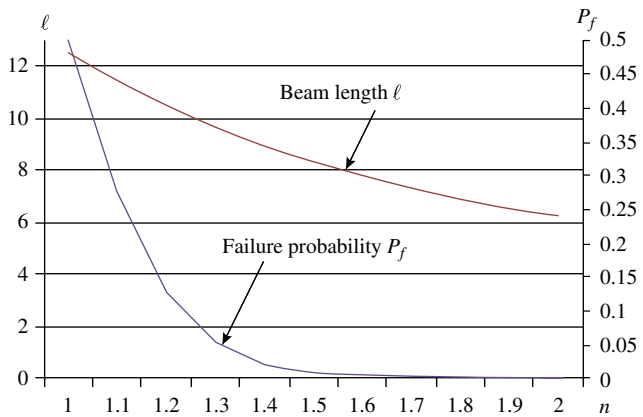


FIGURE 10.4

Influence of the safety factor on beam length and failure probability.

and

$$d = \mu_d \pm 0.25\sigma_d = 1 \pm 0.25(0.1) \text{ in.}$$

in which a 3σ and a 0.25σ tolerance is assumed respectively for these two variables.

The absolute-worst-case design calls for a worst-case scenario, which in this case implies determining the largest possible bending stress and smallest possible material strength, and using both to carry out design of the beam. In this example, we assume that the length of the beam is to be determined using the worst-case scenario. As before, the standard deviation of the beam length is assumed to be 0.1 of its mean value, $\sigma_\ell = 0.1\mu_\ell$. The worst-case scenario for the beam can be identified as

$$S_y = \mu_{S_y} - 3\sigma S_y = 50 - 3(6) = 32 \text{ ksi}$$

and

$$d = \mu_d + 0.25\sigma_d = 1 + 0.25(0.1) = 1.025 \text{ in.}$$

The beam length can be determined by equating the largest stress with the smaller strength:

$$s = \frac{32(p\ell)}{\pi d^3} = S_y$$

Therefore, the beam length can be calculated as

$$\ell = \frac{\pi d^3 S_y}{32p} = \frac{\pi 1.025^3 (32{,}000)}{32(393)} = 8.609 \text{ in}$$

From Eqs 10.11c and 10.12, we have for length $\ell = 8.609$ in.

$$z = \frac{-(50 - 4\mu_\ell)}{\sqrt{6^2 + (0.4\mu_\ell)^2}} = \frac{-(50 - 4(8.609))}{\sqrt{6^2 + (0.4(8.609))^2}} = -2.25$$

and the failure probability $P_f = \Phi(-2.25) = 0.0122 = 1.22\%$. That is, 1.22% of the beam will fail even if we impose a worst-case design scenario. After all, the absolute-worst-case design is not so absolute. On the other hand, if a failure rate greater than 1.22% can be tolerated, for example, if the required minimum failure rate is 5%, the worst-case design is in fact yielding an overdesigned cantilever beam.

10.2.3 SHORT SUMMARY

Several important points were made in the cantilever beam example. First, increasing the safety factor reduces failure probability. However, employing the safety factor approach for design does not guarantee zero failure. Second, failure probability is also reduced by tightening manufacturing tolerance, which narrows the variation of geometric dimensions. Third, in addition to narrowing the variation of random variables, desired failure probability can be achieved by adjusting the mean value of the random variable. Finally, the absolute-worst-case design is not absolute. Failure will still occur. In engineering design, the main objective is often carrying out a trade-off between failure probability and a reasonable product cost.

It is critical to bring probability into product design to adequately address the uncertainties in physical parameters that affect product performance. Design engineers must understand the

probabilistic nature of physical problems and be able to take uncertainties into consideration in the various design phases.

In the following sections, we introduce the subject of reliability analysis systematically and with more rigor. A short review of the fundamentals of engineering statistics is provided next for readers to sharpen their statistics skills before turning to the more sophisticated reliability analysis methods in Section 10.4.

10.3 BASICS OF STATISTICS AND PROBABILISTIC THEORY

Before discussing reliability analysis methods, we briefly review a few basic subjects in statistics and probabilistic theory. This is not a thorough review; instead, only the basics and those topics relevant to the reliability analysis methods in Section 10.4 are included.

10.3.1 EVENTS AND BASIC PROBABILITY RULES

In the following paragraphs, basic terms are defined and basic probability rules are described.

A sample space Ω is a set of all possible outcomes of an experiment. An event E is a subset of the sample space of an experiment. Thus, for example, the sample space of rolling a die is $\Omega = \{1, 2, 3, 4, 5, 6\}$. An event of an even number can be defined as $E_1 = \{2, 4, 6\}$. The probability of event E_1 is obviously $P(E_1) = ½$. Also, as mentioned in the previous section, a failure mode of a structure (or a failure event in statistics) can be modeled as $E_2 = \{S_y \leq s\}$, where s and S_y are the maximum stress and yield strength, respectively. The probability of failure is the probability of event E_2; i.e., $P_f = P(E_2) = P(S_y \leq s) = P(S_y - s \leq 0)$.

If a system is modeled by a number of failure events, failure of the system can be calculated by a union or an intersection of the individual failure events. For a series system, any individual failure event causes system failure. Therefore, the failure of the system is a union of all individual events:

$$E = E_1 \cup E_2 \cup \ldots \cup E_m = \cup_{i=1}^{m} E_i \tag{10.15}$$

where m is the number of individual events. A parallel system does not fail unless all individual events fail simultaneously. Therefore, the failure of the system is an intersection of all individual events:

$$E = E_1 \cap E_2 \cap \ldots \cap E_m = \cap_{i=1}^{m} E_i \tag{10.16}$$

Disjoint, or mutually exclusive, events are defined as

$$E_1 \cap E_2 = \emptyset \tag{10.17}$$

where ø is an empty (or impossible) event. For rolling a die, the two events $E_3 = \{2\}$ and $E_4 = \{6\}$ are disjoint since it is impossible to have both 2 and 6 at the same time when a die is rolled.

There are three fundamental axioms:

Axiom 1: For any event E, $0 \leq P(E) \leq 1$
Axiom 2: For a sample space Ω, $P(\Omega) = 1$
Axiom 3: For mutually exclusive events, $E_1, E_2, \ldots, E_m$, $P(\cup_{i=1}^{m} E_i) = \sum_{i=1}^{m} P(E_i)$

Note that mutually exclusive events are not independent. *Mutually exclusive* and *independent* are two unrelated terms in statistics.

To determine whether events are independent, we have to first understand conditional probability. Conditional probability of an event E_2 given another event E_1 is defined by

$$P(E_2|E_1) = \frac{P(E_2 \cap E_1)}{P(E_1)} \tag{10.18a}$$

Thus,

$$P(E_2 \cap E_1) = P(E_2|E_1)P(E_1) \tag{10.18b}$$

For example, what is the probability that the total of two dice is greater than 8, given that the first die is a 6?

Let us define event $E_1 = \{6\}$ and for the time being, define event E_2, which is the total of two dice greater than 8 regardless of the first die. Then $E_2 = \{(3,6), (4,5), (4,6), (5,4), (5,5), (5,6), (6,3), (6,4), (6,5), (6,6)\}$. There are overall 36 ($6 \times 6 = 36$) possible outcomes when we roll two dice, so the probability of event E_2 is

$$P(E_2) = \frac{10}{36} = \frac{5}{18}$$

Also, $E_1 \cap E_2 = \{(6,3), (6,4), (6,5), (6,6)\}$, so $P(E_1 \cap E_2) = \frac{4}{36} = \frac{1}{9}$. We know that the answer to this question; i.e., the probability that the total of two dice is greater than 8, given that the first die is a 6, is 2/3 because only when the second die is 3, 4, 5, or 6 (4 out of a possible 6) is the total of two dice greater than 8. Therefore, $P(E_2|E_1) = \frac{4}{6} = \frac{2}{3}$. Let us see if Eq. 10.18a gives us the right answer:

$$P(E_2|E_1) = \frac{P(E_2 \cap E_1)}{P(E_1)} = \frac{1/9}{1/6} = \frac{2}{3}$$

It does.

If event E_2 is statically independent of E_1—that is, $P(E_2) = \frac{5}{18}$ regardless if the first die is 6—then

$$P(E_2|E_1) = P(E_2) \tag{10.19}$$

Therefore, in general, if E_1 and E_2 are statically independent, from Eq. 10.18b we have

$$P(E_2 \cap E_1) = P(E_2)P(E_1) \tag{10.20}$$

Again, independent events are not disjoint and vice versa.

The total probability theorem states that if $E_1, E_2, \ldots, E_m$ are mutually exclusive, then, for an event A, we have

$$\begin{aligned} P(A) &= P(A|E_1)P(E_1) + P(A|E_2)P(E_2) + \ldots + P(A|E_m)P(E_m) \\ &= P(A \cap E_1) + P(A \cap E_2) + \ldots + P(A \cap E_m) \\ &= \sum_{i=1}^{m} P(A|E_i)P(E_i) \end{aligned} \tag{10.21}$$

Since

$$P(A \cap E_i) = P(A|E_i)P(E_i)$$

we have

$$P(E_i|A) = \frac{P(A|E_i)P(E_i)}{P(A)} = \frac{P(A|E_i)P(E_i)}{\sum_{i=1}^{m} P(A|E_i)P(E_i)} \tag{10.22}$$

Eq. 10.22 is an extended form of the famous and powerful Bayes' theorem (or law or rule), which is known as the law of total probability.

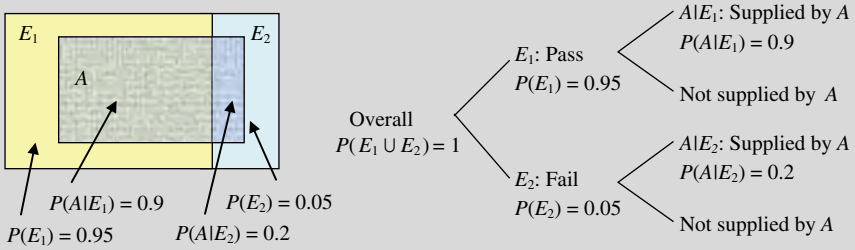
Bayes' theorem provides the actual probability of an event given the measured test probabilities. It has been applied to many problems in engineering and other disciplines for solving problems encountered in, for example, correction for measurement errors and relating actual probability to measured test probability. The following example illustrates the application of the theorem to a simple engineering problem.

EXAMPLE 10.1

Consider steel beams that are tested before use. Let A denote the event that the beams are supplied by vendor A; let E_1 denote the event that beams pass the test; and let E_2 denote the event that beams fail the test. E_1 and E_2 are mutually exclusive. Let the pass rate be 95%—that is, $P(E_1) = 0.95$, and then $P(E_2) = 0.05$. For those beams that pass the test, 90% are supplied by vendor A (i.e., $P(A|E_1) = 0.9$), and for those that fail, 20% are from vendor A (i.e., $P(A|E_2) = 0.2$). What is the probability of beams supplied by vendor A passing the test? In other words, what is $P(E_1|A)$? Also, what percentage of the beams are supplied by vendor A? In other words, what is $P(A)$?

Solution

The answers to these questions can be found using Bayes' theorem as shown in Eq. 10.20. Before we apply the equation to a solution, we analyze the problem so we have a better understanding of the approach to be taken. The problem can be analyzed using either the rectangle or the tree diagram shown in the figures below (left: rectangle diagram; right: tree diagram).



From the figures, we see that the percentage of beams that pass the test and are supplied by vendor A is $P(E_1 \cap A) = P(A|E_1)P(E_1) = (0.9)(0.95) = 0.855$. The number that fail the test and are supplied by vendor A is $P(E_2 \cap A) = P(A|E_2)P(E_2) = (0.2)(0.05) = 0.01$. Hence, the percentage of beams supplied by vendor A is $P(E_1 \cap A) + P(E_2 \cap A) = 0.855 + 0.01 = 0.865 = P(A)$. The probability of the beams supplied by vendor A passing the test is $0.855/(0.855 + 0.01) = 0.988$. The above analysis can be summarized in one equation as follows:

$$P(E_1|A) = \frac{P(E_1 \cap A)}{P(E_1 \cap A) + P(E_2 \cap A)} = \frac{P(A|E_1)P(E_1)}{P(A|E_1)P(E_1) + P(A|E_2)P(E_2)}$$

$$= \frac{0.9 \times 0.95}{0.9 \times 0.95 + 0.2 \times 0.05} = 0.988$$

which is exactly the application of Bayes' theorem shown in Eq. 10.22.

10.3.2 RANDOM VARIABLES AND DISTRIBUTION FUNCTIONS

In this subsection, we discuss random variables and probabilistic distributions that are essential to reliability analysis.

10.3.2.1 Random Variables

A random variable is one whose value is determined by the outcome of a random experiment. In this chapter, X denotes a random variable and x denotes a value of the random variable in an experiment, which represents an event that is a subset of the sample space. In general, a random variable takes on various values x within the range $-\infty < x < \infty$.

Random variables are of two types: discrete and continuous. A discrete random variable is one whose set of assumed values is countable (i.e., arises from counting). A continuous random variable is one whose set of assumed values is uncountable (i.e., arises from measurement). Most random variables encountered in engineering design are continuous—for example, material strength is obtained from measurement instead of counting. Therefore, the discussion in this chapter focuses on continuous random variables.

10.3.2.2 Distribution Functions

The function $f_X(x)$ is called a probability density function (PDF) for the continuous random variable where the total area under the PDF curve bounded by the x-axis, as shown in Figure 10.5(a), is equal to 1:

$$\int_{-\infty}^{\infty} f_X(x)dx = 1 \tag{10.23}$$

The cumulative distribution function of a continuous random variable X, shown in Figure 10.5(b), is defined as

$$F_X(x) = \int_{-\infty}^{x} f_X(s)ds \tag{10.24}$$

If $F_X(x)$ is continuous, the probability of X having a value between a and b can be calculated as

$$F_X(b) - F_X(a) = \int_{-\infty}^{b} f_X(x)dx - \int_{-\infty}^{a} f_X(x)dx = \int_{a}^{b} f_X(x)dx \tag{10.25}$$

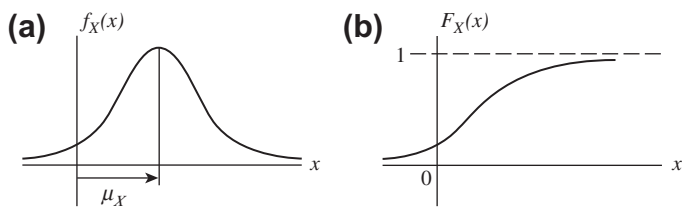


FIGURE 10.5

Distribution functions of a continuous random variable X: (a) PDF and (b) CDF.

Equation 10.25 is illustrated in Figure 10.6, in which $F_X(b) - F_X(a)$ is the area under the curve bounded by the x-axis between a and b. If the random variable X is continuous and if the first derivative of the CDF exists, then the PDF $f_X(x)$ is given by the first derivative of the CDF, $F_X(x)$:

$$f_X(x) = \frac{dF_X(x)}{dx} \tag{10.26}$$

10.3.2.3 Mean Value and Standard Deviation

The mean value (or expected value or average) of a random variable X with PDF $f_X(x)$ is defined as

$$\mu_X = E(X) = \int_{-\infty}^{\infty} x\, f_X(x)dx \tag{10.27}$$

The variance σ_X^2 of X, which is a measure of how far a set of numbers is spread out, is defined as

$$\sigma_X^2 = E\left[(X - \mu_X)^2\right] = \int_{-\infty}^{\infty} (x - \mu_X)^2 f_X(x)dx \tag{10.28}$$

where σ_X is the standard deviation of X. The coefficient of variation of a random variable X indicates the relative amount of uncertainty or randomness, which is defined as

$$V_X = \frac{\sigma_X}{\mu_X} \tag{10.29}$$

10.3.2.4 Joint Probability Density Function

Joint probability expresses the probability that two or more random variables will exist simultaneously. In general, if there are n random variables, the outcome is an n-dimensional vector of them. For example,

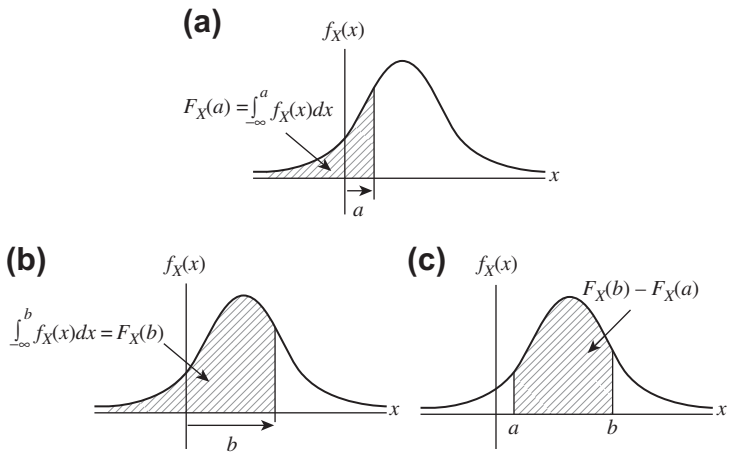


FIGURE 10.6

Value of cumulative distribution function $F_X(x)$: (a) $F_X(a)$, (b) $F_X(b)$, and (c) $F_X(b) - F_X(a)$.

the probability of a two-dimensional case, in which the vector of random variables is $\boldsymbol{X} = [X, Y]^{\mathrm{T}}$, can be calculated as

$$P(a < X < b, c < Y < d) = \int_{c}^{d}\int_{a}^{b} f_{XY}(x, y)dxdy \tag{10.30}$$

where $f_{XY}(x,y)$ is the joint probability density function of the random variables X and Y.

If two random variables X and Y are correlated, X can be affected by the value taken by Y. One of the best ways to visualize the possible relationship is to plot the (X,Y) pair that is produced by several trials of the experiment. An example of correlated samples is shown in Figure 10.7, in which the correlation coefficients ρ (to be discussed next) are 0.5 and -0.5, respectively.

For correlated random variables X and Y, the covariance defined in Eq. 10.31a can be used as a measure to describe a linear association between the two random variables:

$$\begin{aligned} C_{XY} &= \mathrm{Cov}(X, Y) = E[(X - \mu_X)(Y - \mu_Y)] \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x - \mu_X)(y - \mu_Y) f_{XY}(x, y)dxdy \end{aligned} \tag{10.31a}$$

$\mathrm{Cov}(X,Y)$ can also be written as

$$\begin{aligned} \mathrm{Cov}(X, Y) &= E[(X - \mu_X)(Y - \mu_Y)] \\ &= E[XY] - \mu_X\mu_Y \end{aligned} \tag{10.31b}$$

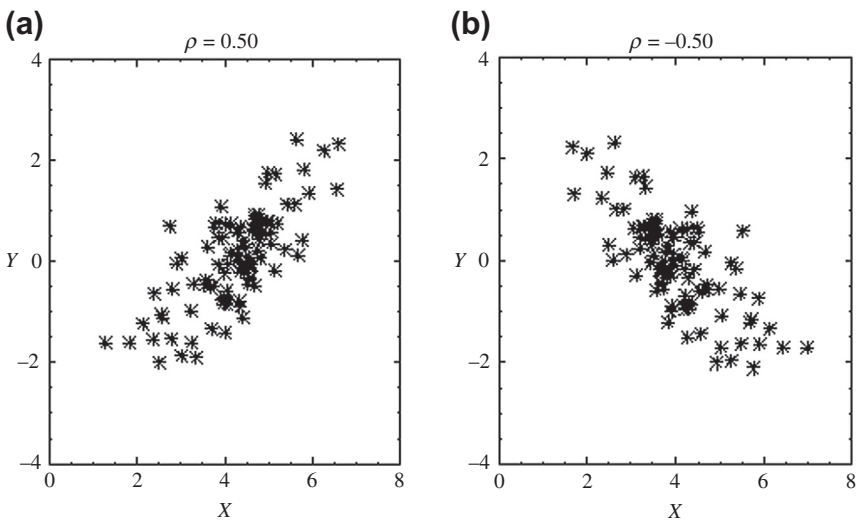


FIGURE 10.7

Correlated random variables X and Y: (a) X and Y correlated with a positive correlation coefficient, $\rho = 0.5$; (b) X and Y correlated with a negative correlation coefficient, $\rho = -0.5$.

The correlation coefficient ρ mentioned previously is a nondimensional measure of the correlation, which is defined as

$$\rho = \frac{C_{XY}}{\sigma_X \sigma_Y} = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y},\ -1 \le \rho \le 1 \tag{10.32}$$

where σ_X and σ_Y are the standard deviations of the random variables X and Y, respectively.

If X and Y are statistically independent (that is, there is no linear relationship between them), the variables are uncorrelated and the correlation coefficient $\rho = 0$. $\rho > 0$ indicates a positive relationship between X and Y; that is, Y increases as X increases. For example, $\rho = 0.5$, shown in Figure 10.7(a), reveals such a correlation. $\rho < 0$ indicates a negative relationship between X and Y; that is, Y decreases as X increases, as depicted in Figure 10.7(b). $\rho = 1$ indicates a perfect positive linear relationship between X and Y; Y linearly increases as X increases. $\rho = -1$ gives a perfect negative linear relationship between X and Y (also called anticorrelation); Y linearly decreases as X increases.

Similarly, for a vector of random variables $\boldsymbol{X} = [X_1, X_2, \ldots, X_n]^{\mathrm{T}}$ with joint PDF $f_{\boldsymbol{X}}(\boldsymbol{x})$, the elements in the vectors of expected values and the covariance matrix are, respectively, $\mu_i = E[X_i]$, $i = 1, n$; and $C_{ij} = \text{Cov}(X_i, X_j)$, $i, j = 1, n$, which is defined as

$$\begin{aligned} C_{ij} &= \text{Cov}\left(X_i, X_j\right) = E\left[\left(X_i - \mu_i\right)\left(X_j - \mu_j\right)\right] \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x_i - \mu_i)\left(x_j - \mu_j\right) f_{X_i X_j}(x_i, x_j)\, dx_i dx_j \end{aligned} \tag{10.33a}$$

Equation. 10.33a can also be written in a matrix form as

$$\begin{aligned} \mathbf{C} &= \begin{bmatrix} E\left[(X_1 - \mu_1)^2\right] & \cdots & \text{Symmetric} \\ E[(X_2 - \mu_2)(X_1 - \mu_1)] & E\left[(X_2 - \mu_2)^2\right] & \cdots \\ & \cdots & \\ E[(X_n - \mu_n)(X_1 - \mu_1)] & \cdots & E\left[(X_n - \mu_n)^2\right] \end{bmatrix} \\ &= \begin{bmatrix} C_{11} & \cdots & & & \text{Symmetric} \\ C_{21} & C_{22} & \cdots & & \\ & & \cdots & & \\ C_{n1} & & & \cdots & C_{nn} \end{bmatrix}_{n \times n} \end{aligned} \tag{10.33b}$$

where

$C_{ii} = \sigma_i^2$, where σ_i is the standard deviation of the random variable X_i.

The correlation coefficient between X_i and X_j is

$$\rho_{ij} = \frac{C_{ij}}{\sigma_i \sigma_j};\ i,j = 1,n;\ -1 \le \rho_{ij} \le 1 \tag{10.34}$$

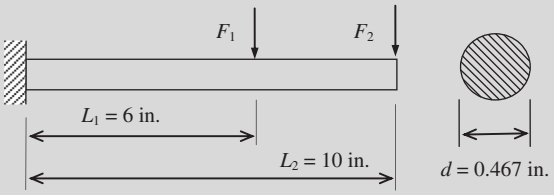
The following example illustrates how random variables can be correlated and how the correlation using correlation coefficient ρ_{ij} can be characterized.

EXAMPLE 10.2

Consider a cantilever beam of solid circular cross-section, as shown below. The diameter of the beam is $d = 0.467$ in. The beam is loaded with two random forces F_1 and F_2 at $L_1 = 6$ in. and $L_2 = 10$ in. respectively. The mean values and standard deviations of these two random forces are, respectively, $\mu_1 = 20$ lb, $\sigma_1 = 4$ lb; and $\mu_2 = 10$ lb, $\sigma_2 = 2$ lb. The shear force V and maximum bending stress s at the root of the beam are, respectively,

$$V = F_1 + F_2, \text{and } s = \frac{32M}{\pi d^3} = \frac{32(F_1L_1 + F_2L_2)}{\pi d^3} = 100\big(F_1L_1 + F_2L_2\big)$$

Calculate the mean value and standard deviation of the maximum bending stress s. If the forces F_1 and F_2 are independent, what is the correlation coefficient between V and s?



Solution

The mean value and standard deviation of the shear force are, respectively,

$$\mu_V = \mu_1 + \mu_2 = 20 + 10 = 30 \text{ lb}$$

and

$$\begin{aligned}
\sigma_V^2 &= E\Big[(V - \mu_V)^2\Big] \\
&= E\Big[((F_1 + F_2) - (\mu_1 + \mu_2))^2\Big] \\
&= E\Big[((F_1 - \mu_1) + (F_2 - \mu_2))^2\Big] \\
&= E\Big[(F_1 - \mu_1)^2 + 2(F_1 - \mu_1)(F_2 - \mu_2) + (F_2 - \mu_2)^2\Big] \\
&= E\Big[(F_1 - \mu_1)^2\Big] + E\Big[(F_2 - \mu_2)^2\Big] \\
&= \sigma_1^2 + \sigma_2^2 \\
&= 4^2 + 2^2 \\
&= 20
\end{aligned}$$

Therefore, the standard deviation of the shear force is

$$\sigma_V = \sqrt{20} = 4.472 \text{ lb}$$

The mean value and standard deviation of the maximum bending stress are, respectively,

$$\mu_s = 100(\mu_1 L_1 + \mu_2 L_2) = 100(20 \times 6 + 10 \times 10) = 22{,}000 \text{ psi}$$

EXAMPLE 10.2—cont'd

and

$$\begin{aligned}
\sigma_s^2 &= E\left[(s-\mu_s)^2\right] \\
&= E[(100(F_1L_1+F_2L_2)-100(\mu_1L_1+\mu_2L_2))^2] \\
&= E[(100(L_1(F_1-\mu_1)+L_2(F_2-\mu_2)))^2] \\
&= E[(10{,}000(L_1^2(F_1-\mu_1)^2+2L_1L_2(F_1-\mu_1)(F_2-\mu_2)+L_2^2(F_2-\mu_2)^2)] \\
&= 10{,}000\left\{L_1^2E\left[(F_1-\mu_1)^2\right]+2L_1L_2E[(F_1-\mu_1)(F_2-\mu_2)]+L_2^2E\left[(F_2-\mu_2)^2\right]\right\} \\
&= 10{,}000\left(L_1^2\sigma_1^2+2L_1L_2\mathrm{Cov}(F_1,F_2)+L_2^2\sigma_2^2\right) \\
&= 10{,}000\left(6^2\times4^2+0+10^2\times2^2\right)=9{,}760{,}000
\end{aligned}$$

Therefore, the standard deviation of the maximum bending stress is

$$\sigma_s=\sqrt{9{,}760{,}000}=3{,}124 \text{ psi}$$

The correlation coefficient between V and s can be found as follows:

$$\begin{aligned}
C_{Vs} &= \mathrm{Cov}(V,s)=E[(V-\mu_V)(s-\mu_s)]=E[Vs]-\mu_V\mu_s \\
&= E[(F_1+F_2)100(F_1L_1+F_2L_2)]-(\mu_1+\mu_2)100(\mu_1L_1+\mu_2L_2) \\
&= 100\left\{L_1E\left[F_1^2\right]+L_2E\left[F_1^2\right]+(L_1+L_2)E[F_1+F_2]-L_1\mu_1^2-L_2\mu_2^2-(L_1+L_2)\mu_1\mu_2\right\} \\
&= 100\left\{L_1E\left[F_1^2\right]+L_2E\left[F_2^2\right]-L_1\mu_1^2-L_2\mu_2^2\right\} \\
&= 100\left\{L_1\left(E\left[F_1^2\right]-\mu_1^2\right)+L_2\left(E\left[F_2^2\right]-\mu_2^2\right)\right\} \\
&= 100\left\{L_1\sigma_1^2+L_2\sigma_2^2\right\} \\
&= 100\left\{6\times4^2+10\times2^2\right\} \\
&= 13{,}600
\end{aligned}$$

$$\rho_{Vs}=\frac{C_{Vs}}{\sigma_V\sigma_s}=\frac{13{,}600}{4.472\times3{,}124}=0.973$$

Hence, the covariance matrix $\boldsymbol{C}$ is

$$\boldsymbol{C}=\begin{bmatrix}1 & 0.973\\ 0.973 & 1\end{bmatrix}$$

10.3.3 PROBABILISTIC DISTRIBUTIONS

In this subsection, we discuss a few common distribution function types, specifically, probability density functions and cumulative distribution functions. We include normal, lognormal, and Weibull distributions (Weibull is one of the extreme value distributions). These are commonly found in engineering applications and are used in the examples in Section 10.4, where we discuss reliability analysis methods.

10.3.3.1 Normal Distribution

A normal distribution function, shown in Figure 10.8, reveals a bell-shape curve. The normal distribution of a random variable X with expected value μ_X and standard deviation σ_X is denoted $N(\mu_X,\sigma_X)$ or $X \sim N(\mu_X,\sigma_X)$, and its PDF is defined as

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma_X} e^{-\frac{1}{2}\left(\frac{x-\mu_X}{\sigma_X}\right)^2}, \quad -\infty < x < \infty \tag{10.35a}$$

The CDF of X is

$$F_X(x) = \Phi\left(\frac{x-\mu_X}{\sigma_X}\right) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma_X} e^{\frac{1}{2}\left(\frac{s-\mu_X}{\sigma_X}\right)^2} ds \tag{10.35b}$$

where $\Phi(\bullet)$ is the standard (or normalized) normal distribution function with mean value 0 and standard deviation 1, represented as N(0,1).

Therefore, a random variable X of normal distribution, denoted $X \sim N(\mu_X,\sigma_X)$, can be normalized by a simple transformation $z = \frac{x-\mu_X}{\sigma_X}$, in which Z is a random variable of standard normal distribution (i.e., $Z \sim N(0,1)$). The cumulative distribution function of the random variable Z of standard normal distribution is thus given by

$$F_X(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma_X} e^{-\frac{1}{2}\left(\frac{s-\mu_X}{\sigma_X}\right)^2} ds = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}} dt = \Phi(z) \tag{10.36}$$

EXAMPLE 10.3

Continue with Example 10.2. If the two random forces F_1 and F_2 are normally distributed, what is the probability that the bending stress s exceeds 24,000 psi?

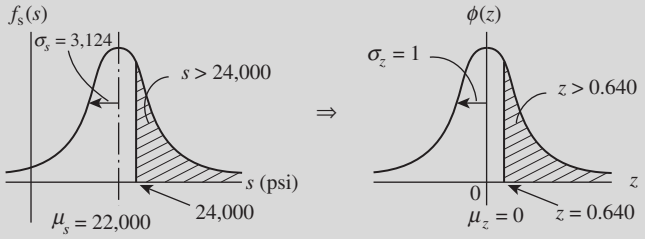
Solution

From Example 10.2, we have the mean value and standard deviation of the bending stress: $\mu_s = 22{,}000$ and $\sigma_s = 3{,}124$ psi. First, we calculate z by

$$z = \frac{s-\mu_s}{\sigma_s} = \frac{24{,}000 - 22{,}000}{3{,}124} = 0.640$$

The probability can then be obtained as (see figure below)

$$P(s > 24{,}000) = P(z > 0.640) = 1 - \Phi(0.640) = 1 - 0.739 = 0.261 = 26.1\%$$



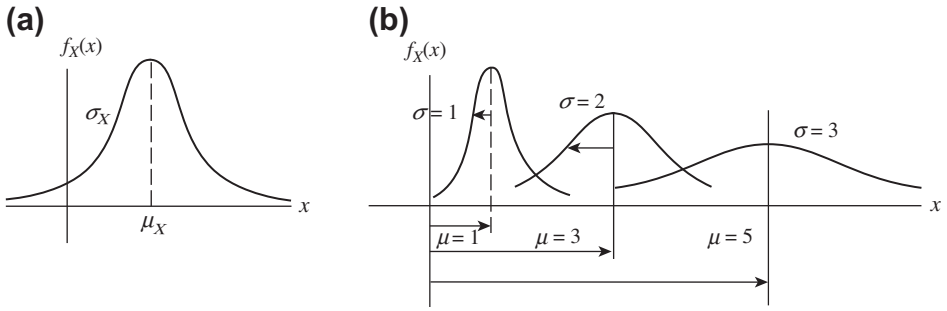


FIGURE 10.8

Normal distribution function $f_X(x)$: (a) $f_X(x)$ with mean value μ_X and standard deviation σ_X; (b) $f_X(x)$ with several mean values and standard deviations.

10.3.3.2 Lognormal Distribution

Lognormal distribution plays an important role in probabilistic design because negative values of engineering phenomena are sometimes physically impossible. Typical uses of lognormal distribution are found in descriptions of fatigue failure, failure rates, and other phenomena involving a large range of data.

The lognormal distribution of a random variable X with expected value μ_X and standard deviation σ_X is denoted LN(μ_X,σ_X) and is defined as

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma_Y} e^{-\frac{1}{2}\left(\frac{\ln(x)-\mu_Y}{\sigma_Y}\right)^2}, 0 < x < \infty \tag{10.37a}$$

in which $f_X(x)$ is the PDF of the random variable X, and

$$\sigma_Y = \sqrt{\ln\left(\left(\frac{\sigma_X}{\mu_X}\right)^2 + 1\right)} \tag{10.37b}$$

and

$$\mu_Y = \ln(\mu_X) - \frac{1}{2}\sigma_Y^2$$

are the standard deviation and expected value for the normal distribution variable $y = \ln(x)$. A few lognormal distribution functions are shown in Figure 10.9. The cumulative distribution function of a lognormal distribution is given as

$$F_X(x) = \int_{-\infty}^{\ln(x)} \frac{1}{\sqrt{2\pi}\sigma_Y} e^{-\frac{1}{2}\left(\frac{s-\mu_Y}{\sigma_Y}\right)^2} ds \tag{10.37c}$$

10.3.3.3 Extreme Value Distributions

Extreme value distributions are used to represent the maximum or minimum of a number of samples of various distributions. There are three types, described in the following paragraphs.

Type 1, also called the Gumbel distribution, is a distribution of the maximum or minimum of a number of samples of normally distributed data. A Gumbel distribution function is defined as

$$f_X(x) = ae^{-e^{-a(x-b)}}e^{-a(x-b)}, -\infty < x < \infty, a > 0 \tag{10.38a}$$

where a and b are scale and location parameters, respectively. The cumulative distribution function of a Gumbel distribution is given as

$$F_X(x) = e^{-e^{-a(x-b)}}, -\infty < x < \infty, a > 0 \tag{10.38b}$$

Type 2, also called the Frechet distribution, is defined as

$$f_X(x) = \frac{k}{v}\left(\frac{v}{x}\right)^{k+1} e^{-\left(\frac{v}{x}\right)^k}, 0 < x < \infty, \kappa \geq 2 \tag{10.39a}$$

The cumulative distribution function of a Frechet distribution is given as

$$F_X(x) = e^{-\left(\frac{v}{x}\right)^k}, 0 < x < \infty, \kappa \geq 2 \tag{10.39b}$$

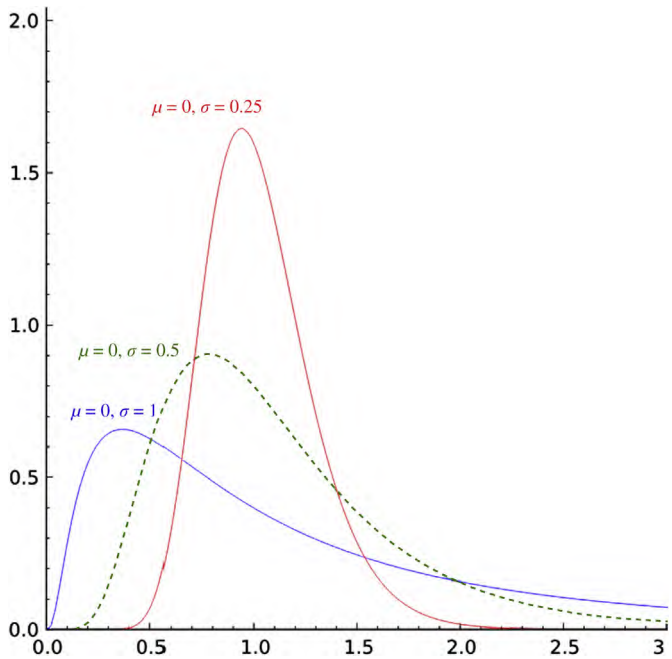


FIGURE 10.9

Lognormal distribution function f_X with several mean values and standard deviations.

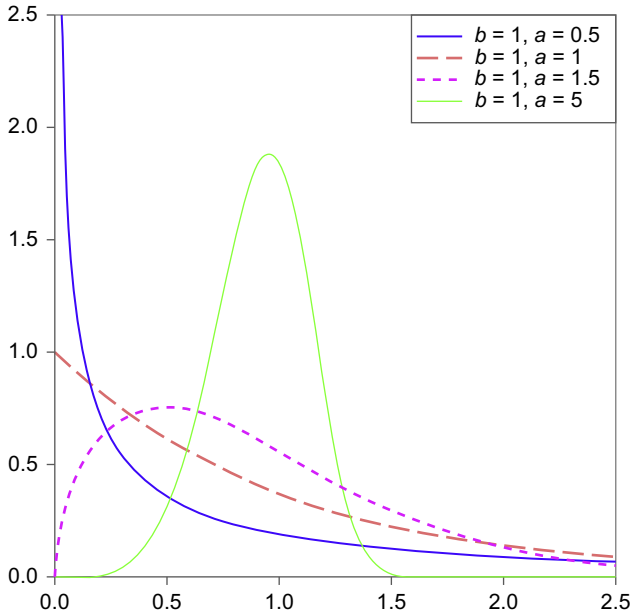


FIGURE 10.10

Weibull distribution function f_X with several mean values and standard deviations.

Type 3, also called the Weibull distribution, is well suited to describing the weakest-link phenomenon, a situation where there are competing flaws contributing to failure. It is often used to describe fatigue, fracture of brittle materials, and strength in composites. The distribution of wind speeds at a given location on earth can also be described with this distribution, which is defined as

$$f_X(x) = \frac{ax^{a-1}}{b^a} e^{-\left(\frac{x}{b}\right)^a}, 0 \le x, a > 0, b > 0 \tag{10.40a}$$

The CDF of a Weibull distribution is given as

$$F_X(x) = 1 - e^{-\left(\frac{x}{b}\right)^a}, 0 \le x, a > 0, b > 0 \tag{10.40b}$$

A few Weibull distribution functions are shown in Figure 10.10.

10.4 RELIABILITY ANALYSIS METHODS

We have reached the key topic of this chapter, reliability analysis methods. Our focus is on those that are the most commonly employed: Monte Carlo simulation, the first-order reliability method (FORM), the second-order reliability method (SORM), importance sampling, and the response surface method. In addition, we discuss the transformation of random variables into standard normal

distribution that is necessary to employ some of these methods. We assume a single failure mode in a product component in this discussion. Multiple failure modes or system failure are discussed in Section 10.5.

10.4.1 THE LIMIT STATE FUNCTION

The first step in reliability analysis is to identify the failure mode in which performance exceeds design limit—that is, how the product or component fails. The mode must be written in a mathematical form that involves random variables. Such a mathematical representation is called the *limit state function*.

A vector of random variables $\boldsymbol{X} = [X_1, X_2, \ldots, X_n]^{\mathrm{T}}$ with joint probability density function $f_{\boldsymbol{X}}(\boldsymbol{x})$ can be used to model the uncertainties of a physical problem. For a structural problem, $\boldsymbol{X}$ can be used to model uncertainties in loads, yield strength, geometric dimensions, material properties, and so forth. Realization of $\boldsymbol{X} = [X_1, X_2, \ldots, X_n]^{\mathrm{T}}$ is denoted $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$, which is a point in the n-dimensional space. A limit state function of these variables can be written as

$$g(\boldsymbol{x}) = g(x_1, x_2, \ldots, x_n) = 0 \tag{10.41}$$

which divides the $\boldsymbol{x}$-space into two regions, safe region R_s and failure region R_f:

$$g(\boldsymbol{x}) = \begin{cases} > 0, x \in R_s \\ \leq 0, x \in R_f \end{cases} \tag{10.42}$$

If in the limit state function $\boldsymbol{x}$ is replaced by random variables $\boldsymbol{X}$, the so-called safety margin M is defined as

$$M = g(\boldsymbol{X}) \tag{10.43}$$

where M is a random variable. The probability of failure P_f of a structure with this failure mode is thus

$$P_f = P(M \leq 0) = P(g(\boldsymbol{X}) \leq 0) = \int_{g(\mathbf{x}) \leq 0} f_{\boldsymbol{X}}(\boldsymbol{x}) d\mathbf{x} = \int_{R_f} f_{\boldsymbol{X}}(\boldsymbol{x}) d\mathbf{x} \tag{10.44}$$

and the reliability of the structure can be simply obtained as $P_R = 1 - P_f$.

The probability integration in Eq. 10.44 is visualized with a two-dimensional case in Figure 10.11(a), which shows the joint PDF $f_{\boldsymbol{X}}(\boldsymbol{x})$ and its contour projected on the x_1-x_2-plane. All points on the projected contours have the same values of $f_{\boldsymbol{X}}(\boldsymbol{x})$ or the same probability density. The limit state function $g(\boldsymbol{x}) = 0$ is also shown. The failure probability P_f is the volume underneath the surface of the joint PDF $f_{\boldsymbol{X}}(\boldsymbol{x})$ in the failure region $g(\boldsymbol{x}) \leq 0$. To show the integration more clearly, the contours of the joint PDF $f_{\boldsymbol{X}}(\boldsymbol{x})$ and the limit state function $g(\boldsymbol{x}) = 0$ are plotted on the x_1-x_2-plane, as shown in Figure 10.11b.

The direct evaluation of the probability integration of Eq. 10.44 is very difficult if not impossible. First, the integration is multidimensional since often multiple random variables are involved in engineering problems. Second, the joint PDF $f_{\boldsymbol{X}}(\boldsymbol{x})$ is in general a nonlinear function. Third, the limit state function $g(\boldsymbol{x})$ is often nonlinear without an analytical form, in which case a numerical method, such as finite element analysis (FEA), is employed for a solution. For this reason, methods other than direct integration have been developed, which are discussed next.

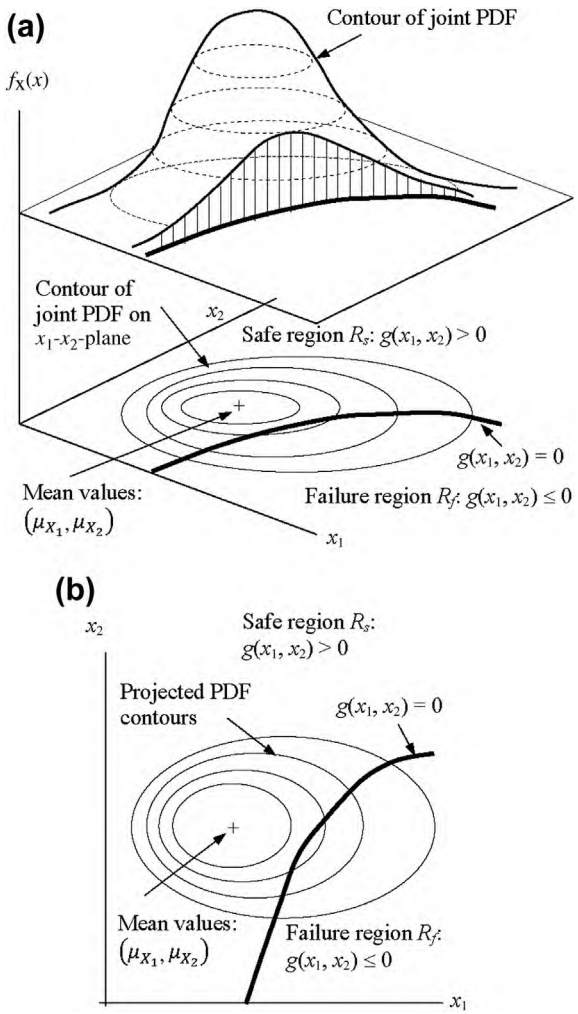


FIGURE 10.11

Probability integration illustrated using a two-dimensional example: (a) isometric view and (b) projected view on the x_1-x_2-plane.

10.4.2 MONTE CARLO SIMULATION

A powerful statistical analysis tool that has been widely used in both engineering and nonengineering applications—*Monte Carlo simulation*, or, simply, *simulation*—is the simplest and most reliable analysis method among many others. After a set of random variables is identified and a

failure mode and associated limit state function are defined, Monte Carlo simulation involves three major steps:

Step 1: Sampling on random input variables $\boldsymbol{X}$.
Step 2: Evaluating the limit state function $g(\boldsymbol{x})$.
Step 3: Statistical analysis of the outcome of the limit state function.

We assume that the CDFs of these respective random variables are known. To simplify our discussion, we further assume that these random variables are independent. Note that Monte Carlo simulation is not limited to problems of independent random variables.

The purpose of sampling the input random variables is to generate samples that represent distributions of the input random variables from their respective CDFs $F_{X_i}(x_i)$, i = 1, n. For each random variable, a set of random variable values $z = [z_1, z_2, \ldots, z_m]^T$ between 0 and 1 is generated first. Note that m is the prescribed number of sample points. These samples $z \in [0, 1]$ are then transformed into sample values of random variable X_i following a given CDF $F_{Xi}(x_i)$ by

$$x_{ij} = F_{X_i}^{-1}(z_j),\ j = 1, m \tag{10.45}$$

where $F_{X_i}^{-1}(z_j)$ is the inverse of the CDF of the ith random variable X_i. For example, if the random variable X_i is normally distributed with $N(\mu_{xi}, \sigma_{xi})$, then

$$z_j = F_{Xi}(x_{ij}) = \Phi\left(\frac{x_{ij} - \mu_{Xi}}{\sigma_{Xi}}\right),\ j = 1, m \tag{10.46a}$$

Thus,

$$x_{ij} = \mu_{Xi} + \sigma_{Xi}, \Phi^{-1}(z_j),\ j = 1, m \tag{10.46b}$$

Note that some software, such as MATLAB, generates sample points for x_{ij} directly, in which case the steps just described may not be necessary. The MATLAB function `normrnd(MU, SIGMA, n, m)` returns a set of sample points in an $n \times m$ matrix chosen from the normal distribution with mean value μ and standard deviation σ.

Once the sample values of all random variables are generated, the limit state function $y_j = g(\boldsymbol{x}_j)$ is solved for each sample point $x_j = [x_{1j}, x_{2j}, \ldots, x_{nj}]^T, j = 1, m$ in step 2. Note that $\boldsymbol{y} = [y_1, y_2 \ldots, y_m]^T = [g(\boldsymbol{x}_1, g(\boldsymbol{x}_2), \ldots g(\boldsymbol{x}_m]^T$ is the vector of m limit state function values for the respective m sample points.

After m samples of output $\boldsymbol{y}$ are obtained, a statistical analysis can be carried out to estimate the failure probability (in addition to characteristics of the output such as mean value and standard deviation) using

$$p_f = \frac{1}{m}\sum_{j=1}^{m} I(g(\boldsymbol{x}_j)) = \frac{m_f}{m} \tag{10.47}$$

where $I(\bullet)$ is an indicator function defined as

$$I(g(\boldsymbol{x})) = \begin{cases} 1, & \text{if } g(\boldsymbol{x}) \le 0 \\ 0, & \text{otherwise} \end{cases} \tag{10.48}$$

EXAMPLE 10.4

We employ Monte Carlo simulation to estimate the failure probability of the cantilever beam discussed in Section 10.2. Recall that the stress s and yield strength S_y are two random variables of normal distribution with $s \sim \mathrm{N}(\mu_s, \sigma_s) = (40, 8)$ ksi and $S_y \sim \mathrm{N}(\mu_{sy}, \sigma_{sy}) = (50, 6)$ ksi. The limit state function for the stress failure mode of the beam was defined as

$$g(\boldsymbol{x}) = S_y - s$$

The failure probability calculated was $P_f = 0.159$ in Section 10.2, which is analytical.

Solution

If we simply use ten sample points, $m = 10$, we can use the MATLAB function `normrnd (MU,SIGMA,1,10)` to generate sample points for s and S_y directly, as shown in Table 10.2 (left two columns). Note that with only ten sample points, three values of the g function are less than 0; hence, the failure probability P_f is 0.3. It is obvious that the failure probability calculated using only ten sample points is inaccurate compared with the analytical solution. We need more sample points.

Table 10.2a Sample Points for s, S_y and g Function

s	S_y	$g = S_y - s$
27.9961	60.0879	32.0918
35.6724	42.0859	6.4135
43.7378	48.5422	4.8044
36.8566	45.0047	8.1481
41.2556	51.3974	10.1418
50.2375	37.4567	−12.7808
6.8483	45.6715	−1.1769
50.5365	42.6835	−7.8530
36.1466	60.8851	24.7385
32.7855	48.5281	15.7425

If we increase the number of sample points, for example from ten to 1,000,000 as shown in Table 10.2b, the failure probability becomes 0.1588, which is very close to the analytical solution of 0.159. The table clearly shows that when we increase the number of sample points the failure probability calculated using Monte Carlo simulation approaches the analytical solution.

Table 10.2b Increasing Sample Points to Improve Accuracy of Probability Estimate

m	P_f
10	0.3000
100	0.1300
1000	0.1510
10,000	0.1546
100,000	0.1598
1,000,000	0.1588

Continued

EXAMPLE 10.4—cont'd

Calculating failure probability using Monte Carlo simulation can be implemented using the following MATLAB script.

```
%Matlab script
m=10
s=normrnd(40,8,1,m)
Sy=normrnd(50,6,1,m)
n=0
for i=1:1:m
g=Sy(i)-s(i)
if g < 0
n=n+1
end
end
Pf=n/m
```

and m_f is the number of sample points that yield a nonpositive limit state function. We are essentially counting the number of failures among the total sample points generated.

10.4.3 THE FIRST-ORDER RELIABILITY METHOD

Monte Carlo simulation is straightforward and reliable. However, as demonstrated in Example 10.4, it requires a large number of sample data to ensure an accurate estimate of failure probability. Engineering applications, for example, complex structural problems that require finite element analysis, often require nontrivial computational time. For such applications, Monte Carlo simulation is infeasible because of the substantial computational effort it requires.

A number of methods have been developed to reduce the computational effort involved in Monte Carlo simulation while still offering sufficiently accurate failure probability estimates. These methods aim to provide acceptable estimates for the integral form of failure probability P_f defined in Eq. 10.44, which can be achieved in two important steps:

Step 1: Simplify the joint probability density function $f_{\boldsymbol{X}}(\boldsymbol{x})$ by transforming a given joint probability density function, which may be multidimensional, into a standard normal distribution function of independent random variables of the same dimensions.
Step 2: Approximate the limit state function $g(\boldsymbol{x}) = 0$ by the Taylor series expansion and by keeping the first few terms for approximation. This is mainly to ease calculation of the failure probability; it has nothing to do with analysis of product performance.

If only linear terms are included in the calculation, the method is referred to as first-order reliability method. When the second-order terms are also considered, the method is referred to as second-order reliability method. In this subsection we introduce FORM. SORM is discussed briefly in Section 10.4.4. For both methods, we assume that the random variables are independent and are normally distributed. Transforming dependent random variables of non-normal distribution is discussed in detail in Section 10.4.5.

10.4.3.1 FORM

The space that contains the given set of random variables $\boldsymbol{X} = [X_1, X_2, \ldots, X_n]^{\mathrm{T}}$ is called the X-space. These random variables are transformed into a standard normal space, called U-space, where the transformed random variables $\boldsymbol{U} = [U_1, U_2, \ldots, U_n]^{\mathrm{T}}$ follow the standard normal distribution (that is, with mean value 0 and standard deviation 1). Such a transformation is carried out based on the condition that the CDFs of the random variables remain the same before and after transformation:

$$F_{X_i}(x_i) = \Phi(u_i) \tag{10.49}$$

where $\Phi(\bullet)$ is the CDF of the standard normal distribution. The transformation can be written as

$$u_i = \Phi^{-1}\left(F_{X_i}(x_i)\right) \tag{10.50a}$$

Thus, the transformed random variable U_i can be written as

$$U_i = \Phi^{-1}\left(F_{X_i}(X_i)\right) \tag{10.50b}$$

If the random variables X_i are independent and normally distributed—that is, $F_{X_i}(x_i) = \Phi\left(\frac{x_i-\mu_i}{\sigma_i}\right)$—the transformation, as illustrated in Figure 10.12, can be obtained as

$$u_i = \Phi^{-1}\left(\Phi\left(\frac{x_i-\mu_i}{\sigma_i}\right)\right) = \frac{x_i-\mu_i}{\sigma_i} \tag{10.51}$$

Thus,

$$x_i = \mu_i + \sigma_i\mu_i \tag{10.52}$$

Note that, as shown in Figure 10.12(c), the projected contours of the transformed PDF on the u_1-u_2-plane are circles centered at the origin.

The limit state function is also transformed into U-space as

$$g(\boldsymbol{x}) = g_u(\boldsymbol{u}) = 0 \tag{10.53}$$

The transformed limit state function separates the U-space into safe regions and failure regions, as illustrated in Figure 10.12(b) and (c). After the transformation, the probability integration of Eq. 10.44 becomes

$$P_f = P\left(g_u(\boldsymbol{u}) \le 0\right) = \int_{g_u(\boldsymbol{u})\le 0} \phi(\boldsymbol{u})d\boldsymbol{u} \tag{10.54}$$

Since all random variables $\boldsymbol{U}$ are independent, the joint PDF is the product of the individual PDFs of standard normal distribution:

$$\phi(\boldsymbol{u}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u_i^2} \tag{10.55}$$

Therefore, the probability integration becomes

$$P_f = \iint \cdots \int_{g_u(\boldsymbol{u})\le 0} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u_i^2} du_1 du_2 \ldots du_n \tag{10.56}$$

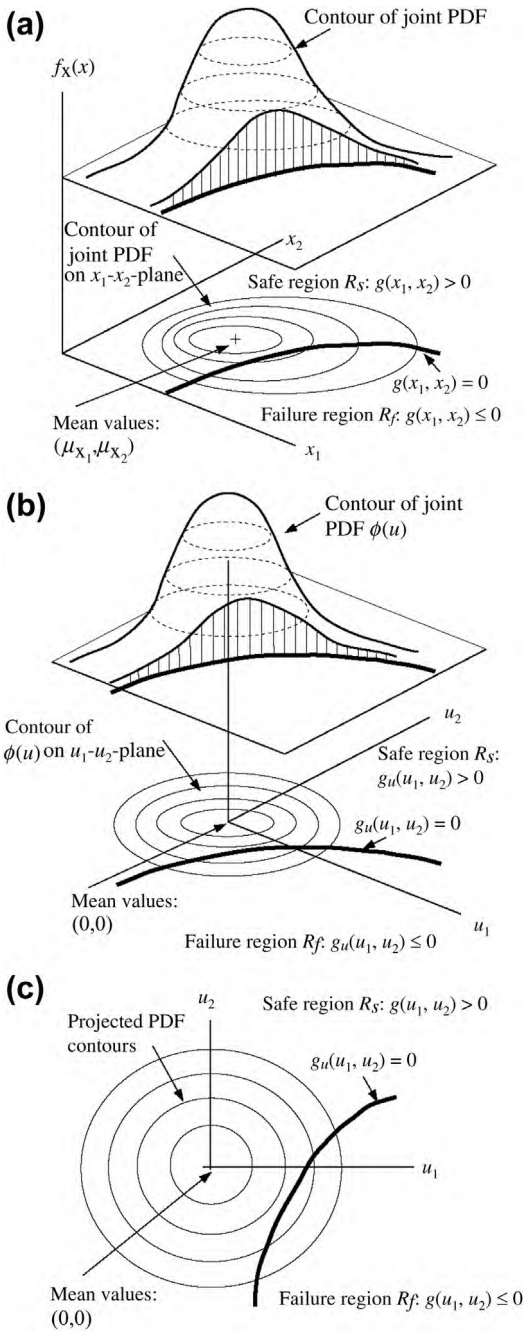


FIGURE 10.12

Transformation of random variables from ***X*** to ***U*** illustrated using a two-dimensional example: (a) PDF and projected contours in *X*-space, (b) standard normal distribution in *U*-space, and (c) projected view on the u_1-u_2-plane.

Although it is obvious that Eq. 10.56 is relatively easier to calculate than Eq. 10.44, it is still difficult, if not entirely impossible, to do so since the limit state function $g_u(\boldsymbol{u})$ is in general a nonlinear function of variable $\boldsymbol{u}$.

If the nonlinearity of the limit state function is not too severe, the integration of Eq. 10.56 can be approximated by

$$P_f \approx \int_{-\infty}^{-\beta} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u_i^2} du = \Phi(-\beta) \tag{10.57}$$

where

Φ is the standard normal distribution function of single dimension.

β is the shortest distance between the origin and the point on the limit state function $g_u(\boldsymbol{u}) = 0$.

The point on the limit state function is depicted as $\boldsymbol{u}^*$ in Figure 10.13 and is called the β-point, design point, or most probable point (MPP). The value of β is referred to as the reliability index.

As illustrated in Figure 10.13(a), the line that connects the origin and the MPP must be perpendicular to the tangent line $L_u(u_1,u_2) = 0$ at the MPP. Since the joint probability density function $\phi(\boldsymbol{u})$ of multidimension is axisymmetric, its projection on the plane that is normal to the tangent line is nothing but a probability density function of the standard normal distribution $\phi(u)$ of a single dimension. The probability integration of Eq. 10.56 can then be approximated by $\Phi(-\beta)$, as stated in Eq. 10.57.

Another way to understand the approximation of Eq. 10.57 is to linearize the limit state function at the MPP—that is, create a tangent line to it, which can be written as

$$g_u(\boldsymbol{u}) \approx g_u(\boldsymbol{u}^*) + \nabla g_u(\boldsymbol{u}^*)(\boldsymbol{u} - \boldsymbol{u}^*) = L(\boldsymbol{u}) \tag{10.58}$$

where $\nabla g_u(\boldsymbol{u}^*)$ is the gradient of function $g_u(\boldsymbol{u}^*)$ defined as

$$\nabla g_u(\boldsymbol{u}^*) = \left[\frac{\partial g_u(\boldsymbol{u})}{\partial u_1}, \frac{\partial g_u(\boldsymbol{u})}{\partial u_2}, \cdots, \frac{\partial g_u(\boldsymbol{u})}{\partial u_n}\right]\Bigg|_{\boldsymbol{u}=\boldsymbol{u}^*} \tag{10.59}$$

Since at the MPP $\boldsymbol{u}^*$, $g_u(\boldsymbol{u}) = 0$, Eq. 10.58 becomes

$$L(\boldsymbol{u}) = \sum_{i=1}^{n} \frac{\partial g_u(\boldsymbol{u})}{\partial u_i}\Bigg|_{u_i=u_i^*} (u_i - u_i^*) = a_0 + \sum_{i=1}^{n} a_i u_i \tag{10.60}$$

where

$$a_0 = -\sum_{i=1}^{n} \frac{\partial g_u(\boldsymbol{u})}{\partial u_i}\Bigg|_{u_i=u_i^*} u_i^* \tag{10.61a}$$

and

$$a_i = \frac{\partial g_u(\boldsymbol{u})}{\partial u_i}\Bigg|_{u_i=u_i^*} \tag{10.61b}$$

Because U_i are random variables of standard normal distribution and $L(\boldsymbol{u})$ is a linear function of the random variables $\boldsymbol{U}$, $L(\boldsymbol{u})$ also has standard normal distribution. The mean value and standard deviation of $L(\boldsymbol{u})$ are, respectively,

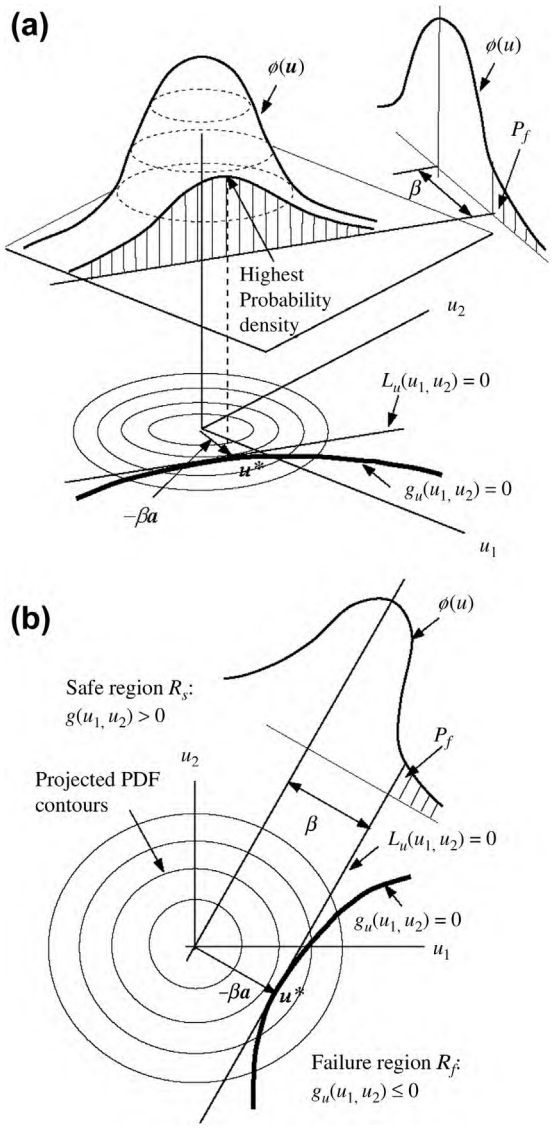


FIGURE 10.13

Approximation of failure probability integration illustrated using a two-dimensional example: (a) PDF and projected contours in *U*-space and (b) projected view on the u_1-u_2-plane.

$$\mu_L = a_0 = -\sum_{i=1}^{n} \left.\frac{\partial g_u(\boldsymbol{u})}{\partial u_i}\right|_{u_i=u_i^*} u_i^* \tag{10.62a}$$

and

$$\sigma_L = \sqrt{\sum_{i=1}^{n} a_i^2} = \sqrt{\sum_{i=1}^{n} \left(\left.\frac{\partial g_u(\boldsymbol{u})}{\partial u_i}\right|_{u_i=u_i^*}\right)^2} \tag{10.62b}$$

Therefore, the probability of failure can be approximated as

$$P_f \approx P(L(\boldsymbol{u}) \le 0) = \Phi\left(\frac{-\mu_L}{\sigma_L}\right) = \Phi\left(\frac{\sum_{i=1}^{n} \left.\frac{\partial g_u(\boldsymbol{u})}{\partial u_i}\right|_{u_i=u_i^*} u_i^*}{\sqrt{\sum_{i=1}^{n} \left(\left.\frac{\partial g_u(\boldsymbol{u})}{\partial u_i}\right|_{u_i=u_i^*}\right)^2}}\right) = \Phi\left(\sum_{i=1}^{n} \alpha_i u_i^*\right) \tag{10.63}$$

where

$$\alpha_i = \frac{\left.\frac{\partial g_u(\boldsymbol{u})}{\partial u_i}\right|_{u_i=u_i^*}}{\sqrt{\sum_{i=1}^{n} \left(\left.\frac{\partial g_u(\boldsymbol{u})}{\partial u_i}\right|_{u_i=u_i^*}\right)^2}}, \text{ and } \boldsymbol{a} = [\alpha_1 \alpha_2, \ldots, \alpha_n]^T = \frac{\nabla g_u(\boldsymbol{u}^*)}{\left\|\nabla g_u(\boldsymbol{u}^*)^T\right\|} \tag{10.64}$$

Then the probability of failure can be written as

$$P_f \approx \Phi\left(\boldsymbol{u}^{*\mathrm{T}} \cdot \boldsymbol{a}\right) \tag{10.65a}$$

As shown in Figure 10.13(b), the position vector of the MPP is $\boldsymbol{u}^* = -\beta \boldsymbol{a}$; hence,

$$P_f \approx \Phi\left(-\beta \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{a}\right) = \Phi(-\beta) \tag{10.65b}$$

since $\boldsymbol{a}$ is a normalized unit vector and $\boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{a} = 1$.

Note that in Eq. 10.63, $P(L(\boldsymbol{u}) \le 0)$ can also be found as

$$P_f \approx P(L(\boldsymbol{u}) \le 0) = P\left(-\beta \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{a} \le 0\right) = \Phi(-\beta) \tag{10.66}$$

in which the failure zone is approximated as $L(\boldsymbol{u}) \le 0$, and $\boldsymbol{a}$, as defined above, is the normalized gradient of the limit state function at the MPP, which is also the gradient of the tangent line that passes the MPP (i.e., $L(\boldsymbol{u}^*) = 0$). Note that in Eq. 10.66 the approximated failure region $L(\boldsymbol{u}) \le 0$ can be represented by any point in the U-space that satisfies $\beta - \boldsymbol{u}^{\mathrm{T}} \cdot \boldsymbol{a} \le 0$, where $\boldsymbol{u}^{\mathrm{T}} \cdot \boldsymbol{a}$ is the projection of the position vector $\boldsymbol{u}$ on vector $\boldsymbol{a}$, which is the normalized gradient at the MPP.

EXAMPLE 10.5

Recall that in the cantilever beam example discussed in Section 10.2, we assumed that both the length and the diameter of the beam cross-section are uncorrelated and are normally distributed—that is, $X_\ell \sim \mathrm{N}(\mu_\ell, \sigma_\ell) = (10, 1)$ in. and $X_d \sim \mathrm{N}(\mu_d, \sigma_d) = (1, 0.1)$ in. The external force p = 393 lb is assumed deterministic. We want to define the maximum bending stress as the failure mode, and calculate the failure probability of the beam using FORM.

Solution

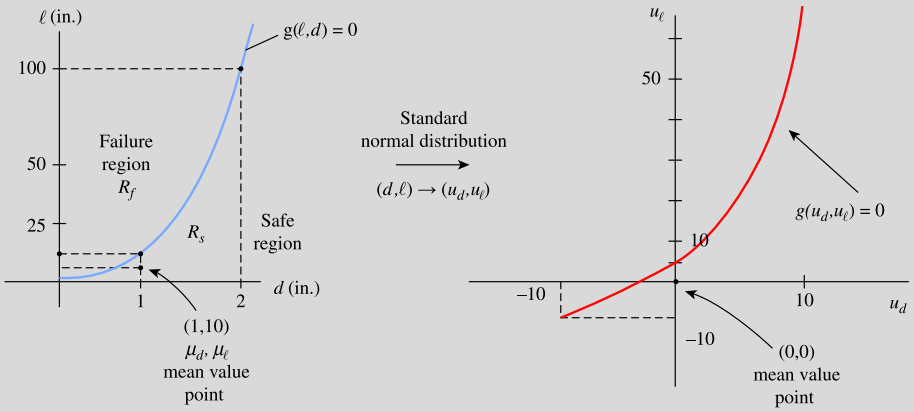
The limit state function for the stress failure mode of the beam can be defined as

$$g(\boldsymbol{x}) = g(d, \ell) = S_y - s = S_y - \frac{32\mathrm{p}\ell}{\pi d^3} = 50 - \frac{4\ell}{d^3} = 0 \ \text{ksi}$$

Rewrite the limit state function (to simplify the math) as

$$g(d, \ell) = 12.5d^3 - \ell = 0$$

Function $g(d, \ell) = 0$ separates the d-ℓ-plane into safe and failure regions, as shown below (left: separation of the d-ℓ plane by $g(d, \ell) = 0$; right: separation of the u_d-u_ℓ plane by $g(u_d,u_\ell) = 0$.



The first step is to transform the random variables $\boldsymbol{X} = [d, \ell]^{\mathrm{T}}$ into $\boldsymbol{U} = [u_d, u_\ell]^{\mathrm{T}}$, in which u_d and u_ℓ are independent random variables of standard normal distribution, using Eq. 10.51:

$$u_d = \frac{d - \mu_d}{\sigma_d} = \frac{d - 1}{0.1} = 10d - 10$$

and

$$u_\ell = \frac{\ell - \mu_\ell}{\sigma_\ell} = \frac{\ell - 10}{1} = \ell - 10$$

Thus,

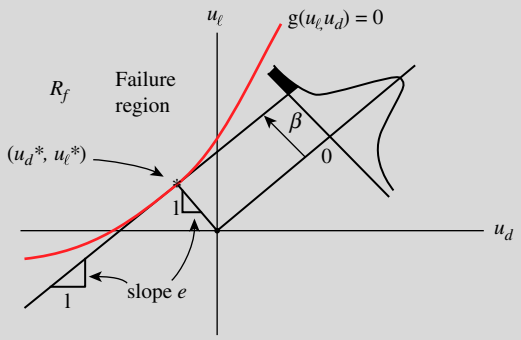
$$g_u(\boldsymbol{u}) = g(u_d, u_\ell) = 12.5(0.1u_d + 1)^3 - (u_\ell + 10) = 0$$

From Eq. 10.59, the gradient of function $g_u(\boldsymbol{u}^*)$ can be calculated as

$$\nabla g_u(\boldsymbol{u}^*) = \left[\frac{\partial g_u(\boldsymbol{u})}{\partial u_d}, \frac{\partial g_u(\boldsymbol{u})}{\partial u_\ell}\right]\Bigg|_{\boldsymbol{u}=\boldsymbol{u}*} = \left[3.75\left(0.1u_d^* + 1\right)^2, -1\right]$$

EXAMPLE 10.5—cont'd

The MPP is located at $\boldsymbol{u}^* = [u_d^*, u_\ell^*]^{\mathrm{T}}$, with a distance β from the origin, as shown below. Location of the MPP at $\boldsymbol{u}^* = [u_d^*, u_\ell^*]^{\mathrm{T}}$ at distance β from the origin.



The straight line that is perpendicular to the tangent line at the MPP is thus

$$\boldsymbol{u}^* = \left[u_d^*, u_\ell^*\right]^{\mathrm{T}} = -\beta\boldsymbol{a} = -\beta[\alpha_1, \alpha_2]^{\mathrm{T}} = -\beta\frac{\nabla g_u(\boldsymbol{u}^*)}{\left\|\nabla g_u(\boldsymbol{u}^*)^T\right\|} = -\beta\frac{\left[3.75\left(0.1u_d^* + 1\right)^2, -1\right]^{\mathrm{T}}}{\sqrt{\left(3.75\left(0.1u_d^* + 1\right)^2\right)^2 + (-1)^2}}$$

from which we obtain the following relationship between u_d^* and u_ℓ^*:

$$\frac{u_\ell^*}{u_{\mathrm{d}}^*} = \frac{-1}{3.75\left(0.1u_d^* + 1\right)^2}$$

u_d^* and u_ℓ^* can be solved by bringing this relation back to the limit state function:

$$g\left(u_d^*, u_\ell^*\right) = 12.5\left(0.1u_d^* + 1\right)^3 - \left(\frac{-u_d^*}{3.75\left(0.1u_d^* + 1\right)^2} + 10\right) = 0$$

Solving for this (using MATLAB for example), we obtain $u_d^* = -0.6553$ and $u_\ell^* = 0.2001$. Hence, the β value can be obtained as

$$\beta = \sqrt{u_d^{*2} + u_\ell^{*2}} = \sqrt{(-0.6553)^2 + (0.2001)^2} = 0.6852$$

and so the failure probability can be approximated, using Eq. 10.66, as

$$P_f \approx \Phi(-\beta) = \Phi(-0.6852) = 0.2466$$

As can be seen, the method for finding the MPP shown in the example is not general. Moreover, this ad hoc approach is not suitable for implementation in computer software. A more general and systematic approach for the MPP search is discussed next.

The key to calculating the failure probability is to locate the MPP in *U*-space. Many numerical methods have been developed for the MPP search. These can be categorized as two types: the reliability index approach (RIA) and the performance measure approach (PMA). RIA employs a forward reliability

analysis algorithm that computes failure probability for a specified performance level in the limit state function. PMA employs an inverse reliability analysis algorithm that computes the response level for a specified failure probability. We introduce both RIA and PMA and use one popular numerical algorithm to illustrate the detailed computation steps they involve. We then discuss the pros and cons of each.

10.4.3.2 The Reliability Index Approach

The problem for an MPP search using RIA can be formulated as follows:

$$\begin{cases} \text{minimize:} & \|\boldsymbol{u}\| \\ \text{subject to:} & g_u(\boldsymbol{u}) = 0 \end{cases} \tag{10.67}$$

in which the MPP is identified by searching a point on the limit state function $g_u(\boldsymbol{u}) = 0$, where the distance between the point to the origin of the U-space is minimum. Again, the distance β is called the reliability index—hence the name of this approach. The RIA was, in fact, illustrated in Example 10.5. Note that in Eq. 10.67, the performance level of the limit state function is specified. As seen in Example 10.5, the yield strength in the limit state function of the stress failure mode is specified as 50 ksi. In this case, the MPP can be searched only for the specified performance level.

One of the most popular algorithms uses a recursive formula and is based on linearization of the limit state function. The MPP search procedure is illustrated in Figure 10.14 using a two-dimensional example, in which the limit state function has been transformed into the U-space.

The basic idea of the algorithm is that it constructs a linear approximation to the limit state function at a search point, calculates the normalized gradients of the limit state function at that search point using Eq. 10.64, and locates the next search point on the limit state function using a vector originating at the origin and pointing in a reverse direction from that of the gradient of the limit state function obtained at the current search point. The distance between the origin and the next search point identified is the β value at the search point. As illustrated in Figure 10.14, an initial search point $\boldsymbol{u}^0$ is usually given as the mean value of the random variables—that is, at the origin O. It is apparent that $\boldsymbol{u}^0$ is most likely not on the limit state function; therefore, $g_u(\boldsymbol{u}^0) \neq 0$.

At this point, the β value is $\beta^0 = 0$. We calculate the normalized gradient of the limit state function at $\boldsymbol{u}^0$ (i.e., $\boldsymbol{a}^0 = \boldsymbol{a}(\boldsymbol{u}^0)$) using Eq. 10.64. We reverse the gradient vector to intersect it with the limit state function; the intersecting point A is assigned as the next search point $\boldsymbol{u}^1$. Note that the distance between the origin and the search point $\boldsymbol{u}^1$ (line OA) is β^1. We calculate the normalized gradient vector of the limit state function at $\boldsymbol{u}^1$ (i.e., $\boldsymbol{a}^1 = \boldsymbol{a}(\boldsymbol{u}^1)$), as shown in Figure 10.14, and move the gradient vector $\boldsymbol{a}^1$ to the origin O, reversing its direction to intersect it with the limit state function; the intersecting point B is assigned as the next search point $\boldsymbol{u}^2$. The process repeats until the search point approaches the MPP $\boldsymbol{u}^*$.

One key step involved in this process is calculating the search point $\boldsymbol{u}^{k+1}$ in the $(k+1)$th iteration by intersecting vector $\boldsymbol{a}^k$ (directed at the origin) with the limit state function. Such a calculation can be extensive when the number of random variables becomes large. One possible way to avoid such calculations is outlined next.

Let us go back to Figure 10.14 and look more closely at iterations 1 and 2—that is, from points A to B. Note that $\boldsymbol{u}^1$ (point A) is on the limit state function (i.e., $g_u(\boldsymbol{u}^1) = 0$) and the length of the line segment OA is β^1. For the current iteration, it is obvious that $\boldsymbol{u}^{1'} = -\beta^1\boldsymbol{a}^1$. If we locate a point $\boldsymbol{u}^{1'} = -\beta^1\boldsymbol{a}^1$ from the origin O (as shown in Figure 10.14, the point is labeled as B'), then $g_u(\boldsymbol{u}^{1'}) \neq 0$.

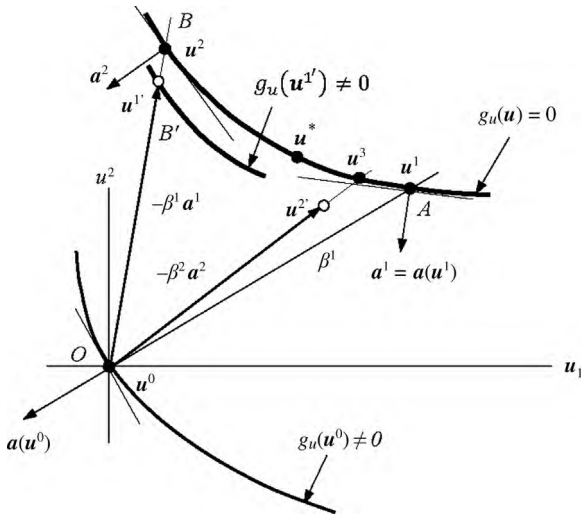


FIGURE 10.14

MPP search using the reliability index approach.

Our goal is to locate $\boldsymbol{u}^2$ (point B) from $\boldsymbol{u}^{1'}$ (point B') without calculating the intersection. First we linearize the limit state function at $\boldsymbol{u}^{1'}$ (as in Eq. 10.58) as

$$L(\boldsymbol{u}) = g_u\left(\boldsymbol{u}^{1'}\right) + \nabla g_u\left(\boldsymbol{u}^{1'}\right)\left(\boldsymbol{u} - \boldsymbol{u}^{1'}\right) \tag{10.68}$$

Eqn. 10.68 is not zero until we evaluate it at $\boldsymbol{u}^2$, which is on the limit function:

$$L\left(\boldsymbol{u}^2\right) = g_u\left(\boldsymbol{u}^{1'}\right) + \nabla g_u\left(\boldsymbol{u}^{1'}\right)\left(\boldsymbol{u}^2 - \boldsymbol{u}^{1'}\right) = 0 \tag{10.69}$$

Note that $\boldsymbol{u}^{1'} = -\beta^1\boldsymbol{a}^1$ and $\boldsymbol{u}^2 = -\beta^2\boldsymbol{a}^1$, where β^2 is the length of line segment OB, which is to be calculated. If we insert these relations into Eq. 10.69, we have

$$g_u\left(\boldsymbol{u}^{1'}\right) + \nabla g_u\left(\boldsymbol{u}^{1'}\right)\left(-\beta^2\boldsymbol{a}^1 + \beta^1\boldsymbol{a}^1\right) = g_u\left(\boldsymbol{u}^{1'}\right) + \nabla g_u\left(\boldsymbol{u}^{1'}\right)\boldsymbol{a}^1\left(\beta^1 - \beta^2\right) = 0 \tag{10.70}$$

Thus,

$$\beta^2 = \beta^1 + \frac{g_u\left(\boldsymbol{u}^{1'}\right)}{\nabla g_u\left(\boldsymbol{u}^{1'}\right)\boldsymbol{a}^1} \tag{10.71}$$

Note that since everything on the right of Eq. 10.71 is known from the first iteration, β^2 can be readily calculated. We may generalize the iteration steps from k to $k+1$, and Eq. 10.71 can be rewritten as

$$\beta^{k+1} = \beta^k + \frac{g_u\left(\boldsymbol{u}^{k'}\right)}{\nabla g_u\left(\boldsymbol{u}^{k'}\right)\boldsymbol{a}^k} \tag{10.72}$$

Therefore,

$$\boldsymbol{u}^{k+1} = -\boldsymbol{a}^k \beta^{k+1} = -\boldsymbol{a}^k \left\{ \beta^k + \frac{g_u\left(\boldsymbol{u}^{k'}\right)}{\nabla g_u\left(\boldsymbol{u}^{k'}\right)\boldsymbol{a}^k} \right\} \tag{10.73}$$

Several other algorithms, such as HL-RF (Hasofer and Lind, 1974; Rackwitz and Fiessler, 1978) and sequential quadratic programming (SQP) (see, for example, Boggs and Tolle, 1995), are also popular for an MPP search.

EXAMPLE 10.6

We illustrate the MPP search algorithm using the cantilever beam from Example 10.5. Recall that the normalized gradient vector of the limit state function is

$$\boldsymbol{a} = \frac{\nabla g_u(\boldsymbol{u}^*)}{\left\|\nabla g_u(\boldsymbol{u}^*)^T\right\|} = \frac{\left[3.75\left(0.1u_d^* + 1\right)^2, -1\right]^T}{\sqrt{\left(3.75\left(0.1u_d^* + 1\right)^2\right)^2 + (-1)^2}}$$

Solution

We start from the mean value point; i.e., $\boldsymbol{u}^{0'} = [0, 0]^T, \nabla g_u(\boldsymbol{u}^{0'}) = [3.75, -1], \boldsymbol{a}^0 = [0.9662, -0.2577]^T$ and $\beta^0 = 0$. From Eqs 10.72 and 10.73, we have

$$\beta^1 = \beta^0 + \frac{g_u\left(\boldsymbol{u}^{0'}\right)}{\nabla g_u\left(\boldsymbol{u}^{0'}\right)\boldsymbol{a}^0} = \frac{2.500}{\left[3.75, -1\right][0.9662, -0.2577]^T} = 0.6442$$

Therefore,

$$\boldsymbol{u}^1 = -\boldsymbol{\alpha}^0\beta^1 = -[0.9662, -0.2577]^T\left(0.6442\right) = [\,-0.6224, 0.1660]^T$$

and

$$\boldsymbol{\alpha}^1 = [0.9570, -0.2902]^T$$

$$\boldsymbol{u}^{1'} = -\beta^1\boldsymbol{\alpha}^1 = -\left(0.6442\right)\left[0.9570, -0.2902\right]^T = \left[\,-0.6164, 0.1869\right]^T$$

The steps are repeated, and the data obtained are listed in Table 10.3. Note, between iterations 2 and 3, the normalized gradient vectors $\boldsymbol{a}$ and β values are identical (up to four digits) so the solution converges. The MPP found is identical to that of Example 10.5. A MATLAB script employed for the computation is shown for reference.

Table 10.3 Numerical Data Obtained for the MPP Search

Iteration	$\boldsymbol{u}$	$\boldsymbol{a}$	β	$\boldsymbol{u}'$
0	0,0	0.9662, −0.2577	0	
1	−0.6224, 0.1660	0.9570, −0.2902	0.6442	−0.6164, 0.1869
2	−0.6556, 0.1986	0.9564, −0.2921	0.6852	−0.6552, 0.2001
3	−0.6553, 0.2001	0.9564, −0.2921	0.6852	

EXAMPLE 10.6—cont'd

```
%Matlab script
% initial iteration (from mean value point)
u = [0,0]
beta = 0
gu = 12.5*(0.1*u(1)+1)^3-(u(2)+10)
vectora = [3.75*(0.1*u(1)+1)^2,-1]
a=normr(vectora)
beta=beta + gu/(vectora*a')
u=-a*beta
%Follow-on iterations
Vectora = [3.75*(0.1*u(1)+1)^2,-1]
a = normr(vectora)
up = -a*beta
gu = 12.5*(0.1*up(1) + 1)^3-(up(2)+10)
Vectora = [3.75*(0.1*up(1)+1)^2,-1]
a = normr(vectora)
beta = beta + gu / (vectora*a')
u = -a*beta
```

10.4.3.3 The Performance Measure Approach

As discussed previously, in the reliability index approach we are searching the MPP along the limit state function where a target performance is prescribed. The point on the limit state function that is the shortest distance to the origin is the MPP. Once the MPP is found, the distance β (reliability index) can be used to approximate the failure probability as $P_f = \Phi(-\beta)$.

The performance measure approach (PMA), on the other hand, is given a target reliability index β (or failure probability P_f and then $\beta = (\Phi^{-1}(P_f))$, and searches for the MPP by bringing the function $g_u(\boldsymbol{u})$ closer to $g_u(\boldsymbol{u}) = 0$, in which the target performance level is achieved. The concept is illustrated in Figure 10.15(a) using a two-dimensional example. The required reliability index β is shown as a circle centered at the origin of the u_1-u_2-plane with radius β. Depending on the $\boldsymbol{u}$ value entered for the MPP search, the limit state function $g_u(\boldsymbol{u})$ is usually nonzero. If the $\boldsymbol{u}$ value entered is in the safe region, then $g_u(\boldsymbol{u}) > 0$. On the other hand, if $\boldsymbol{u}$ is in the failure region, $g_u(\boldsymbol{u}) < 0$. The MPP search becomes one that searches the $\boldsymbol{u}$ to bring the limit state function $g_u(\boldsymbol{u})$ to $g_u(\boldsymbol{u}) = 0$. The problem can be formulated as following

$$\begin{cases} \text{minimize:} & \left|g_u(\boldsymbol{u})\right| \\ \text{subject to:} & \beta = \|\boldsymbol{u}\| \end{cases} \tag{10.74}$$

in which the reliability index $\beta = \Phi^{-1}(P_f)$, and P_f is the required failure probability. Also, in Eq. 10.74 the limit state function $g_u(\boldsymbol{u})$ can be rewritten as

$$g_u(\boldsymbol{u}) = g'(\boldsymbol{u}) - g^t \tag{10.75}$$

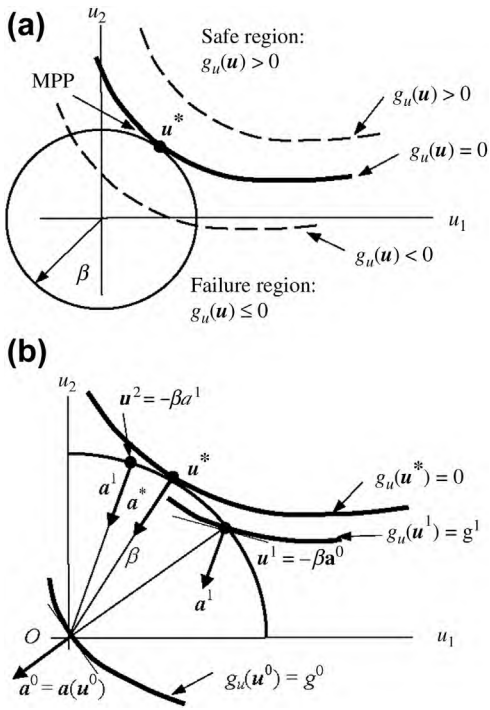


FIGURE 10.15

MPP search using PMA: (a) concept illustration and (b) MPP search scheme.

where

$g'(\boldsymbol{u})$ is the performance measure corresponding to the failure mode.

g^t is the target performance level.

As seen in Example 10.5, the yield strength in the limit state function of the stress failure mode is specified as 50 ksi, which is the target performance level g^t. Note that during the MPP search the search point $\boldsymbol{u}^i$ at the ith iteration yields a nonzero limit state function:

$$e(\boldsymbol{u}^i) = \left| g'(\boldsymbol{u}^i) - g^t \right| > 0 \tag{10.76}$$

where $e(\boldsymbol{u}^i)$ represents the quantity to be minimized to bring the curves shown in Figure 10.15(a) closer to $g_u(\boldsymbol{u}) = 0$, in which the MPP is determined.

We discuss one search algorithm similar to that discussed earlier, using a recursive formula. The algorithm is illustrated in Figure 10.15(b) using a 2D example in which the limit state function has been transformed into the U-space.

Similar to our earlier discussion, an initial search point $\boldsymbol{u}^0$ is usually given as the mean value of the random variables—that is, at the origin O. It is apparent that $\boldsymbol{u}^0$ is most likely not on the limit state function; therefore, $g_u(\boldsymbol{u}^0) = g^0 \neq 0$. At this point, the β value is $\beta^0 = 0$. We calculate the normalized gradient at $\boldsymbol{u}^0$ (i.e., $\boldsymbol{a}^0 = \boldsymbol{a}(\boldsymbol{u}^0)$) using Eq. 10.64. We locate the next search point $\boldsymbol{u}^1$ by simply setting $\boldsymbol{u}^1 = -\beta \mathbf{a}^0$ as shown in Figure 10.15(b). Next we calculate the normalized gradient vector of the limit

state function at $\boldsymbol{u}^1$ (i.e., $\boldsymbol{a}^1 = \boldsymbol{a}(\boldsymbol{u}^1)$) and then locate the next search point $\boldsymbol{u}^2$, again by setting $\boldsymbol{u}^2 = -\beta \mathbf{a}^1$. The process can be generalized for the iterations k to $(k+1)$:

$$\boldsymbol{u}^{k+1} = -\beta \mathbf{a}^k \tag{10.77a}$$

and

$$\mathbf{a}^k = \frac{\nabla g_u\left(\boldsymbol{u}^k\right)^T}{\left\|\nabla g_u\left(\boldsymbol{u}^k\right)\right\|} \tag{10.77b}$$

The process is repeated until the search point approaches the MPP $\boldsymbol{u}^*$.

EXAMPLE 10.7

We illustrate the MPP search using PMA, as discussed earlier, using the same cantilever beam of Example 10.6. Recall that the required reliability index value at the MPP is $\beta = 0.6852$.

Solution

As before, we start from the mean value point, where $\boldsymbol{u}^0 = [0,0]^T$, $g(\boldsymbol{u}^0) = 2.5$, $\nabla g_u(\boldsymbol{u}^0) = [3.75,-1]$, and $\boldsymbol{a}^0 = [0.9662, -0.2577]^T$. From Eq. 10.77a, we have $\boldsymbol{u}^1 = -\beta \boldsymbol{a}^0 = -(0.6852)[0.9662,-0.2577]^T = [-0.6621,0.1766]^T$ and

$$g\left(\boldsymbol{u}^1\right) = 0.0015$$

which is nonzero so the process continues. At the search point $\boldsymbol{u}^1$, we calculate the normalized gradient vector $\boldsymbol{a}^1 = [0.9563, -0.2925]^T$; then

$$\boldsymbol{u}^2 = -\beta \boldsymbol{a}^1 = -\Big(0.6852\Big)[0.9563, -0.2925]^T = [\,-0.6552, 0.2004]^T$$

and

$$g\left(\boldsymbol{u}^2\right) = 0.00006278$$

which is very close to zero so the process terminates. The MPP is found at $\boldsymbol{u}^* = \boldsymbol{u}^2 = [-0.6552,0.2004]^T$, which is very close to the results obtained in Example 10.6.

As illustrated, PMA seems to offer a simpler way to search for the MPP than RIA. It takes fewer calculations in each search iteration since, unlike RIA, it does not need to calculate $\boldsymbol{u}^{i'}$. The advantage of PMA is more significant for reliability-based design optimization (RBDO) in which an objective function (such as structure weight) is to be minimized subject to probabilistic constraints. These probabilistic constraints are often written in terms of reliability index β if FORM is employed for reliability analysis. All probabilistic constraints must be satisfied when an optimal design is found. In each design iteration, they must be evaluated, and those violated must be corrected in successive design iterations.

If RIA is used, Eq. 10.67 must be solved for all probabilistic constraints to see if they are violated by comparing their respective β values obtained at the MPP with the respective target reliability indices. In this case, the MPP search must be carried out for all probabilistic constraints. However, if PMA is used, this search is required only for those that have been violated. This is because a negative value of the limit state function indicates that a failure has occurred, so the sign of the limit state function can be used as a measure to determine whether a probabilistic constraint is satisfied—without actually obtaining a reliability index value by going through the MPP search. For this reason, PMA is more attractive computationally for support of reliability-based design, as reported by Lee et al. (2002). More discussion on RBDO can be found in Chapter 19 Multiobjective Optimization and Advanced Topics.

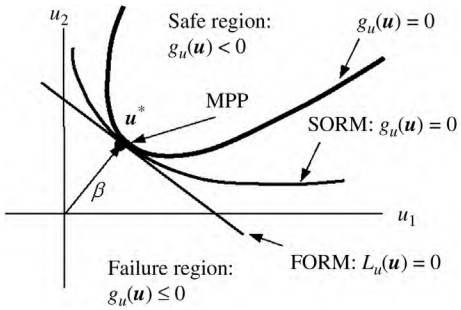


FIGURE 10.16

Comparison of FORM and SORM using a two-dimensional example.

10.4.4 THE SECOND-ORDER RELIABILITY METHOD

For a limit state function of high curvature at the MPP, first-order approximation may not provide adequate accuracy in failure probability calculation. The reason is that the probability integration of Eq. 10.44 (or Eq. 10.56) using FORM ignores the volume underneath the surface of the probability density function and the area between the true limit state function $g_u(\boldsymbol{u}) = 0$ and the linearized function $L_u(\boldsymbol{u}) = 0$. This is illustrated in Figure 10.16.

For limit state functions with a large curvature at the MPP, the second-order reliability method offers a more accurate approximation of failure probability. Instead of discussing the mathematics of SORM, we briefly go over a few key equations to understand how the method works.

SORM uses the second-order Taylor series expansion to approximate the limit state function at the MPP $\boldsymbol{u}^*$. The approximation is given as

$$g_u(\boldsymbol{u}) \approx q(\boldsymbol{u}) = g_u(\boldsymbol{u}^*) + \nabla g_u(\boldsymbol{u}^*)(\boldsymbol{u} - \boldsymbol{u}^*) + \frac{1}{2}(\boldsymbol{u} - \boldsymbol{u}^*)^{\mathrm{T}} \boldsymbol{H}(\boldsymbol{u}^*)(\boldsymbol{u} - \boldsymbol{u}^*) \tag{10.78}$$

where $\boldsymbol{H}(\boldsymbol{u}^*)$ is the Hessian matrix of $n \times n$ at the MPP:

$$\boldsymbol{H}(\boldsymbol{u}^*) = \begin{bmatrix} \frac{\partial^2 g_u}{\partial u_1^2} & \frac{\partial^2 g_u}{\partial u_1 \partial u_2} \cdots & \frac{\partial^2 g_u}{\partial u_1 \partial u_n} \\ \frac{\partial^2 g_u}{\partial u_2 \partial u_1} & \frac{\partial^2 g_u}{\partial u_2^2} \cdots & \frac{\partial^2 g_u}{\partial u_2 \partial u_n} \\ \frac{\partial^2 g_u}{\partial u_n \partial u_1} & \frac{\partial^2 g_u}{\partial u_n \partial u_2} \cdots & \frac{\partial^2 g_u}{\partial u_n^2} \end{bmatrix}_{\boldsymbol{u}^*} \tag{10.79}$$

where n is the number of random variables. Therefore, the probability integration becomes, according to Eq. 10.56,

$$P_f = \iint \cdots \iint_{q_u(\boldsymbol{u}) \le 0} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u_i^2} du_1 du_2 \ldots du_n \tag{10.80}$$

As illustrated in Figure 10.16, a more accurate failure probability estimate, especially for a function of large curvature at the MPP, is provided by integration using the boundary curve $q_u(\boldsymbol{u}) = 0$, which is a better approximation than the straight line $L_u(\boldsymbol{u}) = 0$ used in FORM for the true limit state function $g_u(\boldsymbol{u}) = 0$.

Although the limit state function is simplified using a quadratic function $q_u(\boldsymbol{u})$, Eq. 10.80 is still difficult to calculate, so an approximation has been derived (e.g., Wei, 2006). When β is large enough, an asymptotic solution of the failure probability can be derived as

$$P_f = \Phi(-\beta) \prod_{i=1}^{n-1} \left(\frac{1}{\sqrt{1 + \beta k_i}} \right) \tag{10.81}$$

where k_i is the ith main curvature of the limit state function $g_u(\boldsymbol{u})$ at the MPP.

EXAMPLE 10.8

We calculate the failure probability of the cantilever beam of Examples 10.6 and 10.7 using SORM. Recall that the limit state function of the cantilever beam is

$$g_u(\boldsymbol{u}) = g_u(u_d, u_\ell) = 12.5(0.1u_d + 1)^3 - (u_\ell + 10) = 0$$

The MPP is found previously at $\boldsymbol{u}^* = [-0.6553, 0.2001]^T$, and β is 0.6852.

Solutions

The Hessian matrix can be found using Eq. 10.79 as

$$\boldsymbol{H}(\boldsymbol{u}^*) = \begin{bmatrix} \frac{\partial^2 g_u}{\partial u_d^2} & \frac{\partial^2 g_u}{\partial u_d \partial u_\ell} \\ \frac{\partial^2 g_u}{\partial u_\ell \partial u_d} & \frac{\partial^2 g_u}{\partial u_\ell^2} \end{bmatrix}_{\boldsymbol{u}^*} = \begin{bmatrix} 0.75(0.1u_d + 1) & 0 \\ 0 & 0 \end{bmatrix}_{\boldsymbol{u}^* = [-0.6552, 0.2004]} = \begin{bmatrix} 0.7009 & 0 \\ 0 & 0 \end{bmatrix}$$

The quadratic approximation given in Eq. 10.78 can be obtained as

$$\begin{aligned} q(\boldsymbol{u}) &= g_u(\boldsymbol{u}^*) + \nabla g_u(\boldsymbol{u}^*)(\boldsymbol{u} - \boldsymbol{u}^*) + \frac{1}{2}(\boldsymbol{u} - \boldsymbol{u}^*)^{\mathrm{T}} \boldsymbol{H}(\boldsymbol{u}^*)(\boldsymbol{u} - \boldsymbol{u}^*) \\ &= 0 + [3.275, -1] \begin{bmatrix} u_d + 0.6553 \\ u_\ell - 0.2001 \end{bmatrix} \\ &\quad + \frac{1}{2} [u_d + 0.6553, u_\ell - 0.2001] \begin{bmatrix} 0.7009 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_d + 0.6553 \\ u_\ell - 0.2001 \end{bmatrix} \\ &= 3.275(u_d + 0.6553) - (u_\ell - 0.2001) + \frac{1}{2}(0.7009)(u_d + 0.6553)^2 \\ &= 0.3505u_d^2 + 3.734u_d - u_\ell + 2.490 \end{aligned}$$

Again, bringing this function into Eq. 10.80 for failure probability calculation is very difficult. We use Eq. 10.81 for the calculation instead. For this two-dimensional problem, Eq. 10.81 can be reduced as

$$P_f = \Phi(-\beta) \prod_{i=1}^{n-1} \left(\frac{1}{\sqrt{1 + \beta k_i}} \right) = \Phi(-\beta) \left(\frac{1}{\sqrt{1 + \beta k_1}} \right)$$

Continued

EXAMPLE 10.8—cont'd

The curvature for a two-dimensional curve given implicitly by $g_u(u_d,u_\ell)$ is given by

$$k_1 = \left.\frac{\frac{\partial^2 g_u}{\partial u_d^2}\left(\frac{\partial g_u}{\partial u_\ell}\right)^2 - 2\frac{\partial^2 g_u}{\partial u_\ell \partial u_d}\frac{\partial g_u}{\partial u_d}\frac{\partial g_u}{\partial u_\ell} + \frac{\partial^2 g_u}{\partial u_\ell^2}\left(\frac{\partial g_u}{\partial u_d}\right)^2}{\left(\left(\frac{\partial g_u}{\partial u_d}\right)^2 + \left(\frac{\partial g_u}{\partial u_\ell}\right)^2\right)^{\frac{3}{2}}}\right|_{\boldsymbol{u}^*} = \frac{0.7009(1)^2 - 2(0) + 0}{\left((3.275)^2 + (-1)^2\right)^{\frac{3}{2}}} = 0.0175$$

Thus, the failure probability can be obtained as

$$P_f^{\text{SORM}} = \Phi(-\beta)\left(\frac{1}{\sqrt{1+\beta k_1}}\right) = \Phi(-0.6852)\left(\frac{1}{\sqrt{1+(0.6852)(0.0175)}}\right) = 0.2451$$

Recall that the failure probability calculated using FORM in Example 10.5 is $P_f^{\text{FORM}} = 0.2466$. If we use Monte Carlo simulation for 100,000 sample points, the failure probability $P_f^{\text{MCS}} = 0.2450$. The failure probability calculated using SORM is $P_f^{\text{SORM}} = 0.2451$, which is closer to that of Monte Carlo simulation, so it is considered more accurate than FORM. However, the difference in results obtained from FORM and SORM is not significant because for this problem the curvature of the limit state function at the MPP is small. For cases like this, as illustrated in Figure 10.16, FORM provides an excellent failure probability estimate.

10.4.5 TRANSFORMATION OF RANDOM VARIABLES*

So far in our discussion, we have assumed that all random variables are independent and are normally distributed. They can be easily transformed into those of standard normal distribution using Eqs 10.51 and 10.52. Methods such as FORM and SORM assume random variables that are independent with standard normal distribution.

In engineering applications, random variables are often correlated and reveal non-normal distributions. To employ FORM or SORM for failure probability calculation, these variables must be transformed into those of independent and standard normal distribution. As discussed before, such transformations can be written mathematically (see Eq. 10.50a) as

$$u_i = \Phi^{-1}\left(F_{X_i}(x_i)\right) \tag{10.82}$$

where $\Phi(\bullet)$ is the CDF of the standard normal distribution.

To apply FORM or SORM for failure probability calculation, transformation of random variables is indispensable. It is an important topic that must be addressed carefully. In this subsection, we discuss transformation for three types of random variable: those that are correlated with normal distribution, those that are independent with non-normal distribution, and those that are correlated with non-normal distribution.

10.4.5.1 Correlated Random Variables of Normal Distribution

Let $\boldsymbol{X}$ be a vector of correlated random variables $\boldsymbol{X} = [X_1, X_2, \ldots, X_n]^{\mathrm{T}}$ with joint probability density function $f_{\boldsymbol{X}}(\boldsymbol{x})$ that are of normal distribution. The elements in the vectors of expected values and the

covariance matrix are, respectively, $\mu_i = E[X_i]$, $i = 1, n$, and $C_{ij} = \text{Cov}[X_i, X_j]$, $i, j = 1, n$, which can be written in a matrix form as

$$C_X = \begin{bmatrix} \text{Var}[X_1] & \dots & \dots & \text{Symmetric} \\ \text{Cov}[X_1, X_2] & \text{Var}[X_2] & \dots & \\ \text{Cov}[X_1, X_n] & \text{Cov}[X_2, X_n] & \dots & \text{Var}[X_n] \end{bmatrix}_{n \times n} \tag{10.83}$$

where

$$\begin{aligned} \text{Cov}\left(X_i, X_j\right) &= E\left[\left(X_i - \mu_i\right)\left(X_j - \mu_j\right)\right] \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left(x_i - \mu_i\right)\left(x_j - \mu_j\right) f_{X_i X_j}\left(x_i, x_j\right) dx_i dx_j \end{aligned} \tag{10.84a}$$

and, for $i = j$,

$$\text{Cov}\left(X_i, X_i\right) = E\left[\left(X_i - \mu_i\right)^2\right] = \text{Var}\left[X_i\right] = \sigma_i^2 \tag{10.84b}$$

in which σ_i is the standard deviation of the random variable X_i. The correlation coefficient between X_i and X_j, as discussed in Section 10.3 (Eq. 10.34), is

$$\rho_{ij} = \frac{c_{ij}}{\sigma_i \sigma_j}; \quad i, j = 1, n; \quad -1 \le \rho_{ij} \le 1 \tag{10.85}$$

which can be written in matrix form as

$$\boldsymbol{\rho}_X = \begin{bmatrix} 1 & \dots & \text{Symmetric} \\ \rho_{21} & 1 & \dots \\ & & \dots \\ \rho_{n1} & \rho_{n2} & \dots \quad 1 \end{bmatrix}_{n \times n} \tag{10.86}$$

Transformation of correlated random variables involves two steps:

Step 1: Transform random variables $\boldsymbol{X}$ into $\boldsymbol{Y}$, in which $\boldsymbol{Y} = [Y_1, Y_2, \dots, Y_n]^{\text{T}}$ is a vector of random variables of standard normal distribution (i.e., $Y_i \sim N(1, 0)$ for $i = 1, n$).
Step 2: Transform random variables $\boldsymbol{Y}$ of correlated standard normal distribution into $\boldsymbol{U} = [U_1, U_2, \dots, U_n]^{\text{T}}$, which are random variables of uncorrelated (independent) standard normal distribution.

The step 1 transformation can be carried out as before simply using

$$y_i = \frac{x_i - \mu_{x_i}}{\sigma_{x_i}}, \quad i = 1, n \tag{10.87}$$

It is important to point out that the covariance matrix of $\boldsymbol{Y}$ (that is, $\boldsymbol{C}_Y$, as defined in Eq. 10.83) is equal to the correlation coefficient matrix of $\boldsymbol{X}$ (that is, $\boldsymbol{\rho}_X$, as defined in Eq. 10.88):

$$\boldsymbol{C}_Y = \boldsymbol{\rho}_X \tag{10.88}$$

Proof of Eq. 10.88 is straightforward and therefore left as an exercise.

Step 2 transforms random variables $\boldsymbol{Y}$ of correlated standard normal distribution into $\boldsymbol{U} = [U_1, U_2, \ldots, U_n]^{\mathrm{T}}$, which are random variables of uncorrelated (independent) standard normal distribution. This step can be written as

$$\boldsymbol{Y} = \boldsymbol{T}\boldsymbol{U} \tag{10.89}$$

where $\boldsymbol{T}$ is a lower triangular matrix (i.e., $T_{ij} = 0$ for $j > i$). Using Eq. 10.88, the covariance matrix $\boldsymbol{C}_Y$ for $\boldsymbol{Y}$ can be written as

$$\boldsymbol{C}_Y = E\left[\boldsymbol{Y}\boldsymbol{Y}^{\mathrm{T}}\right] = E\left[\boldsymbol{T}\boldsymbol{U}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{T}^{\mathrm{T}}\right] = \boldsymbol{T}E\left[\boldsymbol{U}\boldsymbol{U}^{\mathrm{T}}\right]\boldsymbol{T}^{\mathrm{T}} = \boldsymbol{T}\boldsymbol{T}^{\mathrm{T}} = \boldsymbol{\rho}_X \tag{10.90}$$

The elements in $\boldsymbol{T}$ can be determined from $\boldsymbol{T}\boldsymbol{T}^{\mathrm{T}} = \boldsymbol{\rho}_X$ as

$$T_{jj} = \sqrt{1 - \sum_{k=1}^{j-1} T_{jk}^2}, j = 1, n \tag{10.91a}$$

and

$$T_{ij} = \frac{\rho_{ji} - \sum_{k=1}^{j-1} T_{jk} T_{ik}}{T_{jj}}, i > j \tag{10.91b}$$

Thus, the transformation from $\boldsymbol{X}$ to $\boldsymbol{U}$ can be obtained in matrix form as

$$\boldsymbol{X} = \boldsymbol{\mu}_{\boldsymbol{X}} + \boldsymbol{D}\boldsymbol{T}\boldsymbol{U} \tag{10.92}$$

where $\boldsymbol{D}$ is a diagonal matrix with standard deviations σ_{Xi} in the diagonal.

Using Eq. 10.92, the limit state function can be written as $g(\boldsymbol{X}) = g_u(\boldsymbol{\mu}_{\boldsymbol{X}} + \boldsymbol{D}\boldsymbol{T}\boldsymbol{U}) = 0$. Methods such as FORM or SORM can then be employed for failure probability calculation. Derivation of Eqs 10.91a and 10.91b is straightforward and is left as an exercise. A simple example illustrates the transformation.

EXAMPLE 10.9

A limit state function is defined as

$$g(\boldsymbol{x}) = x_1 - x_2 + x_3^2$$

which consists of three random variables X_1, X_2, and X_3 of normal distribution with respective mean values and standard deviations $X_1 \sim \mathrm{N}(25,0.25)$, $X_2 \sim \mathrm{N}(4,0.2)$, and $X_3 \sim \mathrm{N}(2,0.1)$. The correlated coefficient of the random variables $\boldsymbol{X} = [X_1, X_2, \ldots, X_n]^T$ is given as

$$\boldsymbol{\rho}_{\boldsymbol{X}} = \begin{bmatrix} 1 & 0.5 & 0.2 \\ 0.5 & 1 & 0.4 \\ 0.2 & 0.4 & 1 \end{bmatrix}_{3\times 3}$$

We want to write the limit state function in terms of independent random variables $\boldsymbol{U}$ of standard normal distribution.

Solution

We start by transforming the random variables $\boldsymbol{X}$ into $\boldsymbol{U}$ as discussed previously. Step 1 involves transforming $\boldsymbol{X}$ into $\boldsymbol{Y}$, which are random variables of standard normal distribution; that is,

$$Y_1 = \frac{X_1 - 25}{0.25}, Y_2 = \frac{X_2 - 4}{0.2}, \text{ and } Y_3 = \frac{X_3 - 2}{0.1}$$

EXAMPLE 10.9—cont'd

As we saw earlier, random variables $\boldsymbol{Y}$ are transformed into $\boldsymbol{U}$ in step 2, following Eqs 10.91a and 10.91b. From the given correlated coefficient matrix $\boldsymbol{\rho}_X$, we have

$$T_{11} = 1, \quad T_{21} = \rho_{12} = 0.5$$

and

$$T_{31} = \rho_{13} = 0.2$$

Then

$$T_{22} = \sqrt{1 - T_{21}^2} = \sqrt{1 - 0.5^2} = 0.866$$

$$T_{32} = \frac{\rho_{23} - T_{21}T_{31}}{T_{22}} = \frac{0.4 - (0.5)(0.2)}{0.866} = 0.346$$

$$T_{33} = \sqrt{1 - T_{31}^2 - T_{32}^2} = \sqrt{1 - 0.2^2 - 0.346^2} = 0.919$$

Thus, the transformation matrix $\boldsymbol{T}$ is

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 0.866 & 0 \\ 0.2 & 0.346 & 0.919 \end{bmatrix}_{3\times 3}$$

Now, since $\boldsymbol{Y} = \boldsymbol{TU}$, we can find $\boldsymbol{U}$ by

$$\boldsymbol{U} = \boldsymbol{T}^{-1}\boldsymbol{Y} = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 0.866 & 0 \\ 0.2 & 0.346 & 0.919 \end{bmatrix}^{-1} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} Y_1 \\ -0.577Y_1 + 1.15Y_2 \\ -0.000251Y_1 - 0.435Y_2 + 1.09Y_3 \end{bmatrix}$$

And we can express $\boldsymbol{X}$ in terms of $\boldsymbol{U}$, using Eq. 10.92:

$$\boldsymbol{X} = \boldsymbol{\mu}_X + \boldsymbol{DTU} = \begin{bmatrix} 25 \\ 4 \\ 2 \end{bmatrix} + \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & 0.866 & 0 \\ 0.2 & 0.346 & 0.919 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix}$$

$$= \begin{bmatrix} 25 + 0.25U_1 \\ 4 + 0.1U_1 + 0.173U_2 \\ 2 + 0.02U_1 + 0.0346U_2 + 0.0919U_3 \end{bmatrix}$$

Thus, the limit state function can be written in $\boldsymbol{U}$ as

$$g_u(\boldsymbol{u}) = (25 + 0.25U_1) - (4 + 0.1U_1 + 0.173U_2) + (2 + 0.02U_1 + 0.0346U_2 + 0.0919U_3)^2$$

10.4.5.2 Independent Random Variables of Non-Normal Distribution

Transformation of independent random variables $\boldsymbol{X}$ of non-normal distribution to random variables $\boldsymbol{U}$ of standard normal distribution is straightforward. Since there is no correlation between X_i and X_j the two can be transformed individually using Eq. 10.50a:

$$x_i = F_{X_i}^{-1}(\Phi(u_i)), i = 1, n$$

Then the limit state function can be written in terms of $\boldsymbol{U}$ as

$$g(\boldsymbol{x}) = g(x_1, x_2, \ldots, x_n) = g_u\left(F_{X_1}^{-1}(\Phi(u_1)), F_{X_2}^{-1}(\Phi(u_2)), \ldots, F_{X_n}^{-1}(\Phi(u_n))\right) = 0$$

For example, if X is a random variable of lognormal distribution (i.e., $X \sim LN(\mu_X, \sigma_X)$), X can be transformed into a random variable U of standard normal distribution following the definition of lognormal distribution shown in Eqs 10.63b and 10.63c. In other words, the cumulative distribution function of a lognormal distribution is found to be

$$F_X(x) = \Phi\left(\frac{\ln(x) - \mu_L}{\sigma_L}\right) \int_{-\infty}^{\ln(x)} \frac{1}{\sqrt{2\pi}\sigma_y} e^{-\frac{1}{2}\left(\frac{s-\mu_L}{\sigma_L}\right)^2} \mathrm{d}s \tag{10.93a}$$

in which

$$\sigma_L = \sqrt{\ln\left(\left(\frac{\sigma_X}{\mu_X}\right)^2 + 1\right)}, \quad \mu_L = \ln(\mu_X) - \frac{1}{2}\sigma_L^2 \tag{10.93b}$$

Thus, the transformation is simply

$$x = F_X^{-1}\left(\Phi\left(u\right)\right) \tag{10.93c}$$

Therefore,

$$u = \frac{\ln(x) - \mu_L}{\sigma_L} \tag{10.94}$$

and

$$x = e^{\mu_L + u\sigma_L} \tag{10.95}$$

For a random variable X with a Gumbel distribution (also called a type 1 extreme value distribution)—$X \sim EXI(\mu_X,\sigma_X)$—the CDF as defined in Section 10.3 (Eq. 10.38b) is

$$F_X(x) = e^{-e^{-a(x-b)}}, -\infty < x < \infty, a > 0 \tag{10.96}$$

where a and b are scale and location parameters, respectively. For

$$a = \frac{\pi}{\sqrt{6\sigma_X}}, \quad b = \mu_X - \frac{0.5772}{a}$$

the random variable X can be transformed into U as follows. First, we equate the CDF of X with a standard normal CDF of U by

$$F_X(x) = e^{-e^{-a(x-b)}} = \Phi(u) \tag{10.97}$$

from which we obtain

$$x = b - \frac{1}{a}\ln(-\ln(\Phi(u))) \tag{10.98}$$

For random variables with a CDF other than lognormal or Gumbel, we follow the same approach discussed earlier for such transformations.

EXAMPLE 10.10

A limit state function is defined as

$$g(\boldsymbol{x}) = x_1 + x_2^2$$

which consists of two random variables X_1 and X_2. X_1 has a lognormal distribution with mean value 20 and standard deviation 5 (i.e., $X_1 \sim LN(20,5)$). X_2 has a Gumbel distribution with mean value 1 and standard deviation 0.1 (i.e., $X_2 \sim EXI(1,0.1)$). These two random variables are independent. We want to write the limit state function in terms of independent random variables $\boldsymbol{U}$ of standard normal distribution.

Solution

Using Eqs 10.95 and 10.98, X_1 and X_2 can be transformed, respectively, into U_1 and U_2:

$$X_1 = e^{\mu_L + u_1 \sigma_L} = e^{2.97 + 0.2462 u_1}$$

where

$$\sigma_L = \sqrt{\ln\left(\left(\frac{\sigma_X}{\mu_X}\right)^2 + 1\right)} = \sqrt{\ln\left(\left(\frac{5}{20}\right)^2 + 1\right)} = 0.2462$$

and

$$\mu_L = \ln(\mu_X) - \frac{1}{2}\sigma_L^2 = \ln(20) - \frac{1}{2}(0.2462)^2 = 2.965$$

$$X_2 = b - \frac{1}{a}\ln(-\ln(\Phi(u_2))) = 0.9550 - 0.07794\ln(-\ln(\Phi(u_2)))$$

where

$$a = \frac{\pi}{\sqrt{6}\sigma_X} = \frac{\pi}{\sqrt{6}\times 0.1} = 12.83, \quad b = \mu_X - \frac{0.5772}{a} = 1 - \frac{0.5772}{12.83} = 0.9550$$

Hence, the limit state function can be written in $\boldsymbol{U}$:

$$g_u(\boldsymbol{u}) = \left(e^{2.97 + 0.2462 u_1}\right) + (0.9550 - 0.07794\ln(-\ln(\Phi(u_2))))^2$$

10.4.5.3 Correlated Random Variables of Non-Normal Distribution

Transformation of correlated random variables of non-normal distribution is more involved than the transformations just discussed. There are two widely used techniques for this: the Rosenblatt transformation (Rosenblatt, 1952) and the Nataf transformation (Nataf, 1962; Liu and Der Kiureghian, 1986). We briefly discuss only the Rosenblatt transformation with an example to provide some basic understanding. Interested readers may refer, for example, to Hurtado (2004) for a more in-depth discussion.

Transformation of correlated random variables $\boldsymbol{X}$ of non-normal distribution to the U-space of uncorrelated random variables $\boldsymbol{U}$ of normalized normal distribution can be defined as

$$\begin{cases} x_1 = F_{X_1}^{-1}(\Phi(u_1)) \\ x_2 = F_{X_2|X_1}^{-1}(\Phi(u_2)|X_1 = x_1) \\ \cdots \\ x_n = F_{X_n|X_1 \ldots X_{n-1}}^{-1}(\Phi(u_n)|X_1 = x_1, \ldots X_{n-1} = x_{n-1}) \end{cases} \tag{10.99}$$

where $F^{-1}_{X_i|X_1\ldots X_{i-1}}(\Phi(u_i)|X_1 = x_1, \ldots X_{i-1} = x_{i-1})$ is the distribution of X_i given $X_1 = x_1, \ldots X_{i-1} = x_{i-1}$. The conditional distribution function $F_{X_i|X_1,\ldots X_{i-1}}(x_1, \ldots, x_{i-1})$ can be written as

$$F_{X_i|X_1,\ldots X_{i-1}}(x_i|x_1, \ldots, x_{i-1}) = \frac{\int_{-\infty}^{x_i} f_{X_1,\ldots X_{i-1}X_i}(x_1, \ldots, x_{i-1}, s)ds}{f_{X_1,\ldots X_{i-1}}(x_1, \ldots, x_{i-1})} \tag{10.100}$$

where $f_{X_1\ldots X_i}(x_1, \ldots, x_i)$ is the joint probability density function of random variables $X_1, \ldots, X_i$.

Note that the probability density functions and cumulative distribution functions $f_{X_1,\ldots X_i}(x_1, \ldots, x_i)$ and $F_{X_i|X_1,\ldots X_{i-1}}(x_1, \ldots, x_{i-1})$ must be calculated for each ith random variable before Eq. 10.99 can be employed for the transformation. The transformation starts by calculating $f_{X_1}(x_1)$ and $F_{X_1}(x_1)$; then x_1 can be obtained as $x_1 = F^{-1}_{X_1}(\Phi(u_1))$. Once x_1 in terms of u_1 is obtained, $F_{X_2|X_1}(x_2|x_1)$ can be calculated using $f_{X_1}(x_1)$ and $f_{X_1,\ldots X_n}(x_1, \ldots, x_n)$ following Eq. 10.100, and then x_2 can be obtained as $x_2 = F^{-1}_{X_2|X_1}(\Phi(u_2)|X_1 = x_1)$. The following example illustrates the computation process.

EXAMPLE 10.11

This example has been slightly modified from that given in Ditlevsen and Madsen (1996). A limit state function is defined as

$$g(\boldsymbol{x}) = 2x_1^2 + \ln(1 + x_2) - (1 + x_1)x_2$$

which consists of two random variables X_1 and X_2. Also, X_1 and X_2 are correlated with a joint distribution function

$$F_{X_1X_2}(x_1, x_2) = 1 - e^{-x_1} + e^{-(x_1+x_2+x_1x_2)}, x_1 > 0, x_2 > 0$$

The joint probability density function $f_{X_1X_2}(x_1, x_2)$ can be obtained by

$$f_{X_1X_2}(x_1, x_2) = \frac{\partial^2 F_{X_1X_2}(x_1, x_2)}{\partial x_1 \partial x_2} = (x_1 + x_2 + x_1x_2)e^{-(x_1+x_2+x_1x_2)}, x_1 > 0, x_2 > 0$$

Solution

We first calculate the probability and distribution functions for these random variables. Those for X_1 can be calculated, respectively, as

$$\begin{aligned} f_{X_1}(x_1) &= \int_0^\infty f_{X_1X_2}(x_1, x_2)\mathrm{d}x_2 = \int_0^\infty (x_1 + x_2 + x_1x_2)e^{-(x_1+x_2+x_1x_2)}dx_2 \\ &= e^{-x_1}\left[x_1\int_0^\infty e^{-(1+x_1)x_2}\mathrm{d}x_2 + (1 + x_1)\int_0^\infty x_2e^{-(1+x_1)x_2}dx_2\right] \\ &= e^{-x_1}\left[\frac{x_1}{1 + x_1} + \frac{1 + x_1}{(1 + x_1)^2}\right] = e^{-x_1} \end{aligned}$$

and

$$F_{X_1}(x_1) = \int_0^{x_1} f_{X_1}(s)ds = \int_0^{x_1} e^{-s}ds = e^{-s}\Big|_0^{x_1} = 1 - e^{-x_1}$$

Those for X_2 given X_1 can be calculated, respectively, as

EXAMPLE 10.11—cont'd

$$f_{X_2}(x_2|x_1) = \frac{f_{X_1X_2}(x_1,x_2)}{f_{X_1}(x_1)} = \frac{(x_1+x_2+x_1x_2)e^{-(x_1+x_2+x_1x_2)}}{e^{-x_1}}$$
$$= (x_1+x_2+x_1x_2)e^{-(1+x_1)x_2}$$

with the corresponding CDF

$$F_{X_2|X_1}(x_2|x_1) = \int_0^{x_2} (x_1+s+x_1s)e^{-(1+x_1)s}ds$$
$$= \int_0^{x_2} \left(x_1e^{-(1+x_1)s} + (1+x_1)se^{-(1+x_1)s}\right)ds = 1-(1+x_2)e^{-(1+x_1)x_2}$$

Following Eq. 10.99, we have

$$F_{X_1}(x_1) = 1-e^{-x_1} = \Phi(u_1)$$

Thus,

$$x_1 = -\ln(1-\Phi(u_1))$$

Now, for X_2, we have

$$F_{X_2|X_1}(x_2|x_1) = 1-(1+x_2)e^{-(1+x_1)x_2} = \Phi(u_2)$$

Hence,

$$\ln(1+x_2)-(1+x_1)x_2 = -\ln(1-\Phi(u_2))$$

Therefore, the limit state function can be written in $\boldsymbol{U}$ as

$$g(\boldsymbol{x}) = 2x_1^2+\ln(1+x_2)-(1-x_1)x_2 = 2(\ln(1-\Phi(u_1)))^2-\ln(1-\Phi(u_2))$$

10.4.6 IMPORTANCE SAMPLING

As shown previously, the computation cost of Monte Carlo simulation is very high, especially for applications of small failure probability, which are common in design engineering. The reason is that only samples that fall into the failure region contribute to the failure probability estimate. If most sample points are taken with a distribution that concentrates in an area in the safe zone and is away from the limit state function, the number of sample points used for Monte Carlo simulation must be very large to obtain an accurate failure probability estimate.

To improve the computational efficiency of Monte Carlo simulation, importance sampling methods are commonly employed. The idea of important sampling is to sample the random variables according to an alternative set of distributions such that more sample points fall into the failure region. As a result, more sample points contribute to the probability estimation and so fewer overall sample points are required for a desired accuracy. This concept is illustrated in Figure 10.17, in which the sampling is based on the MPP. We introduce the MPP-based importance sampling method in this subsection.

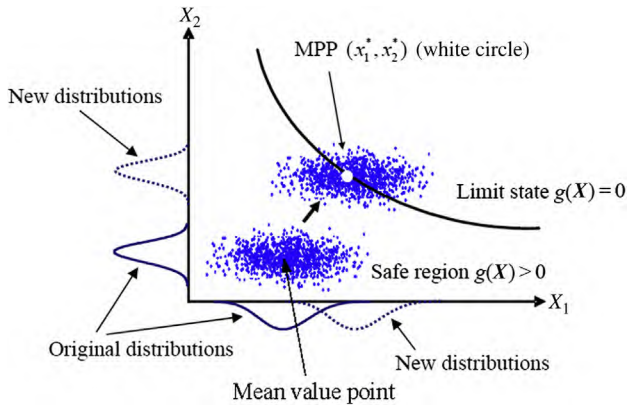


FIGURE 10.17

Importance sampling.

As illustrated in Figure 10.17, sample points generated from the original distribution of random variables X_1 and X_2 fall into the safe region, in which the sampling is often centered on the variables' mean value point. If a new set of distributions of X_1 and X_2 is selected, for example with the mean value point moving to the MPP, many more sample points fall into the failure region. Since more sample points in a given simulation contribute to the failure probability estimate, a more accurate estimate can be expected with a smaller sample set.

After the MPP is obtained, samples are selected around the MPP to evaluate the probability of failure through importance sampling. To do so, an importance sampling density, $h_{\boldsymbol{X}}(\boldsymbol{x})$, is introduced into the Monte Carlo estimation to obtain the probability of failure:

$$P_f = \int_{g(\boldsymbol{x})\le 0} f_{\boldsymbol{X}}(\boldsymbol{x})d\boldsymbol{x} = \int I[g(\boldsymbol{x})]f_{\boldsymbol{X}}(\boldsymbol{x})d\boldsymbol{x} = \int \left\{ I[g(\boldsymbol{x})]\frac{f_{\boldsymbol{X}}(\boldsymbol{x})}{h_{\boldsymbol{X}}(\boldsymbol{x})} \right\} h_{\boldsymbol{X}}(\boldsymbol{x})d\boldsymbol{x} \tag{10.101}$$

where the importance sampling density $h_{\boldsymbol{X}}(\boldsymbol{x})$ is the same as $f_{\boldsymbol{X}}(\boldsymbol{x})$ except that the mean values of $\boldsymbol{X}$ are replaced by the MPP $\boldsymbol{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^{\mathrm{T}}$. For example, if random variable X_1 is normally distributed with $\mathrm{N}(\mu_1,\sigma_1)$ and

$$f_{X_1}(x_1) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{1}{2}\left(\frac{x_1-\mu_1}{\sigma_1}\right)^2}$$

the corresponding importance sampling distribution is $N(x_1^*, \sigma_1)$ and

$$h_{X_1}(x) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{1}{2}\left(\frac{x_1-x_1^*}{\sigma_1}\right)^2}$$

It is noted that the importance sampling density $h_{X_1}(x)$ is centered at x_1^*.

Eqn. 10.101 indicates that the probability of failure is the mean of the integrand $\left\{ I[g(\boldsymbol{x})] \frac{f_X(\boldsymbol{x})}{h_X(\boldsymbol{x})} \right\}$ that is evaluated at the samples of $\boldsymbol{X}$ drawn from the importance sampling density $h_X(\boldsymbol{x})$. Therefore,

$$P_f = \frac{1}{m} \sum_{j=1}^{m} \boldsymbol{I}\left[g\left(\boldsymbol{x}_j\right)\right] \frac{f_X\left(\boldsymbol{x}_j\right)}{h_X\left(\boldsymbol{x}_j\right)} \tag{10.102}$$

In the case of multiple random variables, for example $\boldsymbol{X} = [X_1, X_2, \ldots, X_n]^{\mathrm{T}}$, if they are all normally distributed with $N(\mu_i, \sigma_i)$, then the joint probability density function becomes

$$f_X(\boldsymbol{x}) = \phi(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{C}) = \frac{1}{\sqrt{|\boldsymbol{C}|(2\pi)^n}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{C}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})} \tag{10.103}$$

where $\boldsymbol{\mu}$ is an $n \times 1$ vector of the respective random variables' mean values, and $\boldsymbol{C}$ is the covariance matrix of $n \times n$, which is symmetric and positive definite, as discussed before. In this case, the corresponding importance sampling distribution is $\boldsymbol{N}(\boldsymbol{x}^*, \boldsymbol{\sigma})$ and

$$h_X(\boldsymbol{x}) = \phi(\boldsymbol{x}, \boldsymbol{x}^*, \boldsymbol{C}) = \frac{1}{\sqrt{|\boldsymbol{C}|(2\pi)^n}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^{\mathrm{T}} \boldsymbol{C}^{-1} (\boldsymbol{x}-\boldsymbol{x}^*)} \tag{10.104}$$

We use the same cantilever beam to illustrate the method in Example 10.12.

EXAMPLE 10.12

We use MPP-based importance sampling on the cantilever beam discussed in this subsection. Recall from Example 10.8 that the MPP is found at $\boldsymbol{u}^* = [u_d^*, u_\ell^*]^{\mathrm{T}} = [-0.6552, 0.2004]^{\mathrm{T}}$ and that the diameter and length random variables are $X_\ell \sim \mathrm{N}(\mu_\ell, \sigma_\ell) = (10,1)$ in. and $X_d \sim \mathrm{N}(\mu_d, \sigma_d) = (1,0.1)$ in. respectively. Also, the failure probability estimated using FORM and SORM are, respectively, $P_f^{\mathrm{FORM}} = 0.2466$ and $P_f^{\mathrm{SORM}} = 0.2451$. Note that the MPP is transformed back into the X-space as

$$x^* = [d^*, \ell^*]^{\mathrm{T}} = \left[0.1u_d^* + 1, u_\ell^* + 10\right]^{\mathrm{T}} = [0.9345, 10.2004]^{\mathrm{T}}$$

The limit state function in the X-space is

$$g(\boldsymbol{X}) = g(d, \ell) = 12.5d^3 - \ell$$

It is not necessary to transform the MPP and limit state function back into the X-space for importance sampling. The procedure shown next is applicable to the simulation in U-space as well.

Solution

From Eq. 10.103, the joint probability density function becomes

$$f_X(\boldsymbol{x}) = \frac{1}{\sqrt{|\boldsymbol{C}|(2\pi)^n}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{C}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})} = \frac{1}{\sqrt{(0.01)(2\pi)^2}} e^{-\frac{1}{2}\left(100(d-1)^2 + (\ell-10)^2\right)}$$

where the covariance matrix $\boldsymbol{C}$ is

$$\boldsymbol{C} = \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\ell^2 \end{bmatrix} = \begin{bmatrix} 0.1^2 & 0 \\ 0 & 1^2 \end{bmatrix} = \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix}$$

Continued

EXAMPLE 10.12—cont'd

and

$$h_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{\sqrt{|\boldsymbol{C}|(2\pi)^n}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^T \boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{x}^*)} = \frac{1}{\sqrt{(0.01)(2\pi)^2}} e^{-\frac{1}{2}\left(100(d-0.9345)^2+(\ell-10.2004)^2\right)}$$

Thus, using MPP-based sampling, the failure probability can be estimated as

$$P_f = \frac{1}{m}\sum_{j=1}^{m} I\left[g(\boldsymbol{x}_j)\right]\frac{f_{\boldsymbol{X}}(\boldsymbol{x}_j)}{h_{\boldsymbol{X}}(\boldsymbol{x}_j)} = \frac{1}{m}\sum_{j=1}^{m} I\left[g(\boldsymbol{x}_j)\right]\frac{e^{-\frac{1}{2}\left(100(d_j-1)^2+(\ell_j-10)^2\right)}}{e^{-\frac{1}{2}\left(100(d_j-0.9345)^2+(\ell_j-10.2004)^2\right)}}$$

Table 10.4 shows the simulation results. The left column contains the number of sample points. The middle and right columns contain the failure probability obtained from Monte Carlo simulation and MPP-based importance sampling, respectively. As shown, 1000 sample points gives a reasonably accurate estimate for the failure probability using the MPP-based sampling method. The MATLAB scripts that perform the calculation are shown for reference.

Table 10.4 Numerical Data to Illustrate the Importance Sampling Method

m	P_f (Monte Carlo)	P_f (MPP-Based Sampling)
100	0.2200	0.2165
1000	0.2640	0.2368
10,000	0.2422	0.2429
100,000	0.2444	0.2443
1,000,000	0.2437	

```
% Monte Carlo simulation
m=10000
d=normrnd(1,0.1,1,m)
l=normrnd(10,1,1,m)
n=0
for i=1:1:m
if (12.5*d(i)^3-l(i)) < 0
% MPP-based sampling
m=100000
d=normrnd(0.9345,0.1,1,m)
l=normrnd(10.2004,1,1,m)
n=0
for i=1:1:m
g = 12.5*d(i)^3-l(i)
if g < 0
f = exp(-0.5*(100*(d(i)-1)^2+(l(i)-10)^2))
h = exp(-0.5*(100*(d(i)-0.9345)^2+(l(i)-10.2004)^2))
n = n+ f/h
end
```

10.4.7 THE RESPONSE SURFACE METHOD

The practicality of reliability analysis methods for a specific limit state depends on how complex the formulation of the limit state function is. Often the limit state function is not available in explicit form, but rather defined implicitly through a complicated numerical procedure, given, for example, by finite element analysis. For such limit state formulations, the needed calculations may require prohibitive computational effort.

One way to alleviate such expensive computations is to approximate the limit state surface in a numerical-experimental way by using a surface in explicitly mathematical form and then performing a reliability analysis. This procedure is referred to as the response surface method. In this subsection, we discuss a basic form of this method and use a simple example to illustrate it.

Let $\boldsymbol{X} = [X_1, X_2, \ldots, X_n]^{\mathrm{T}}$ be the vector of n random variables. The central idea of the response surface method is to approximate the exact limit state function $g(\boldsymbol{x})$, which is usually known through an algorithmic procedure, by a polynomial function $\widehat{g}(\boldsymbol{x})$. In practice, quadratic functions are commonly used in the form

$$g(\boldsymbol{x}) \approx \widehat{g}(\boldsymbol{x}) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} a_{ii} x_i^2 + \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} a_{ij} x_i x_j \tag{10.105}$$

where the set of coefficients $\boldsymbol{a} = \{a_0, a_i, a_{ii}, a_{ij}\}$ that correspond to the constant, linear, square, and cross terms, respectively, are to be determined.

A limited number of evaluations of the limit state function are required to build the surface. A reliability analysis can then be performed by means of the analytical expression $\widehat{g}(\boldsymbol{x})$ in Eq. 10.105 instead of the true limit state function $g(\boldsymbol{x})$. This approach is particularly attractive when Monte Carlo simulation is used to obtain reliability results.

The unknown coefficient $\boldsymbol{a}$ is often determined using the least-squares method. After choosing a set of fitting points, $\boldsymbol{x}^k$, $k = 1, m$ (k is the index instead of the power order) for which the exact function values $y^k = g(\boldsymbol{x}^k)$ are calculated using, for example, finite element analysis. An error measure $e(\boldsymbol{a})$, defined by

$$e(\boldsymbol{a}) = \sum_{k=1}^{m} \left(y^k - \widehat{g}\left(\boldsymbol{x}^{\boldsymbol{k}}\right) \right)^2 \tag{10.106}$$

is minimized with respect to $\boldsymbol{a}$. Reformulating Eq. 10.105 in the form

$$\widehat{g}(\boldsymbol{x}) = \left[1, x_i, x_i, x_i x_j\right] \left[a_0, a_i, a_{ii}, a_{ij}\right]^{\mathrm{T}} \equiv \boldsymbol{V}(\boldsymbol{x}) \cdot \boldsymbol{a}^{\mathrm{T}} \tag{10.107}$$

where $i, j = 1, n$, and $j \neq i$, the least-square problem becomes

$$\text{Minimize} \left\{ \sum_{k=1}^{m} \left(y^k - \boldsymbol{V}\left(\boldsymbol{x}^k\right) \cdot \boldsymbol{a}^{\mathrm{T}} \right)^2 \right\} \tag{10.108}$$

After some basic algebra (left as exercise), the solution to the problem yields

$$\boldsymbol{a} = \left(\boldsymbol{v}^{\mathrm{T}} \boldsymbol{v}\right)^{-1} \boldsymbol{v}^{\mathrm{T}} \boldsymbol{y} \tag{10.109}$$

where $\boldsymbol{\nu}$ is the matrix whose rows are the vectors $\boldsymbol{V}(\boldsymbol{x}^k)$ and $\boldsymbol{y}$ is the vector whose components are $y^k = g(\boldsymbol{x}^k)$.

The various response surface methods proposed in the literature differ only in the terms retained in the polynomial expression (Eq. 10.105) and in the selection of the coordinates of the fitting points—that is, the experimental design used in the regression analysis. It is emphasized that $m \geq n'$, where n' is the number of coefficients in $\boldsymbol{a}$, is required to solve Eq. 10.106. Furthermore, the fitting points have to be chosen in a consistent way to obtain independent equations.

EXAMPLE 10.13

Once again we use the cantilever beam discussed in this subsection to illustrate the response surface method. The limit state function in X-space is rewritten as follows (to avoid division by zero):

$$g(\boldsymbol{x}) = g(d, \ell) = 12.5d^3 - \ell = 0 \tag{10.110}$$

Note that if we choose a cubic function as the response surface, we obtain the exact solution. This most likely will not happen since in real applications the limit state function is unknown and is highly non-linear. In this example, we use a quadratic function to create a response surface that approximates the cubic limit state function.

Solution

The response surface using quadratic function can be stated as:

$$\begin{aligned}\widehat{g}(\boldsymbol{x}) &= a_0 + a_1 x_1 + a_2 x_2 + a_{11} x_1^2 + a_{22} x_2^2 + a_{12} x_1 x_2 \\ &= \left[1 \;\; x_1 \; x_2 \; x_1^2 \; x_2^2 \; x_1 x_2\right]\left[a_0 \, a_1 \, a_2 \, a_{11} \, a_{22} \, a_{12}\right]^T = \boldsymbol{V}(\boldsymbol{x}) \cdot \boldsymbol{a}^{\mathrm{T}}\end{aligned}$$

We choose nine arbitrary points to create the response surface:

$$[-1, 1], [0, 1], [1, 1], [-1, 0], [0, 0], [1, 0], [-1, -1], [0, 1], [1, -1]$$

For the first data point $x^1 = [-1,1]$, the vector $\boldsymbol{V}(\boldsymbol{x}^1)$ can be found as

$$\boldsymbol{V}(\boldsymbol{x}^1) = [1, -1, 1, 1, 1, -1]_{1\times 6}$$

We repeat the same for all nine points to form the matrix $\boldsymbol{V}$, which is 9×6. In the meantime, we calculate vector $y^k = g(\boldsymbol{x}^k)$ for them, using Eq. 10.110, as

$$\boldsymbol{y}_{9\times 1} = [-13.5, -1, 11.5, -12.5, 0, 12.5, -11.5, 1, 13.5]^{\mathrm{T}}$$

In practical applications, y^k can be obtained only from numerical solutions—for example using finite element analysis. Solve $\boldsymbol{a}$ using Eq. 10.109 as

$$\boldsymbol{a}_{6\times 1} = [0, 12.5, -1, 0, 0, 0]^{\mathrm{T}}$$

The response surface found is thus

$$\widehat{g}(\boldsymbol{x}) = 12.5x1 - x2$$

which is nothing but a straight line. A straight line is certainly not able to exactly represent a cubic function of the true limit state function. However, the function $\widehat{g}(\boldsymbol{x})$ is obtained in a closed form that can be evaluated very quickly. If we use a different set of data points, $\boldsymbol{x}^k$, we will most likely obtain a different straight line, which may or may not provide a better approximation. Therefore, selecting data points is an important step in using the response surface method.

10.4.8 SHORT SUMMARY

Among the methods discussed, Monte Carlo simulation is certainly the simplest to use, especially for those who have only limited working knowledge of probability and statistics. It can be applied to virtually any performance function and distribution. In addition, it is computationally robust; with a sufficient number of simulations, it can always converge. For reliability analysis, Monte Carlo simulation is generally computationally expensive. The higher the reliability, the larger the simulation size needed. Because of its accuracy, Monte Carlo simulation is widely used in two areas: engineering applications where function evaluations (state limit function) are not computationally expensive, and baseline solutions for validation of other methods. However, its computational inefficiency prevents its regular use for problems where function evaluation is expensive.

Importance sampling is essentially a form of Monte Carlo simulation in which sampling uses a new set of distributions for the random variables—for example, moving the mean value point to the MPP so that many more sample points fall into the failure region. As a result, many fewer sample points are needed for an accurate failure probability estimate. However, this method requires that the MPP first be identified.

MPP search is also an essential step in FORM and SORM. Of the two, FORM is more popular and widely used for engineering applications, especially those requiring finite element analysis for calculating limit state functions. This is because once the MPP is found, the estimation of failure probability is extremely straightforward. However, if the limit state function has high curvature at the MPP, the estimate may not be accurate. SORM improves the accuracy in probability estimate, but it requires that the curvature of the limit state function be calculated. For problems with large numbers of random variables, curvature calculation may be computationally expensive.

The response surface method is also relatively simple to implement. One of its key issues is the selection of sample data for response surface construction. Once a response surface is created, Monte Carlo simulation can be employed in estimating failure probability since a closed-form equation of the response surface is available. If the response surface closely resembles the true limit state function, this method can be very attractive for general applications, in which evaluating the limit state function is computationally expensive.

One possible combination is the use of FORM to provide a first estimate for failure probability and then use importance sampling at the MPP to improve the accuracy of the FORM approximation, if necessary.

10.5 MULTIPLE FAILURE MODES*

So far in our discussion, we have assumed, for the failure probability calculation, one single failure mode in a component. In this section, we address the failure probability calculation for a mechanical system or a component with multiple failure modes. The questions to be answered are how the individual limit states interact with each other and how overall reliability can be estimated.

Series and parallel systems are discussed in Sections 10.5.1 and 10.5.2, respectively. FORM approximation for a series system is introduced in Section 10.5.3, which is brief, providing only basic concepts and methods.

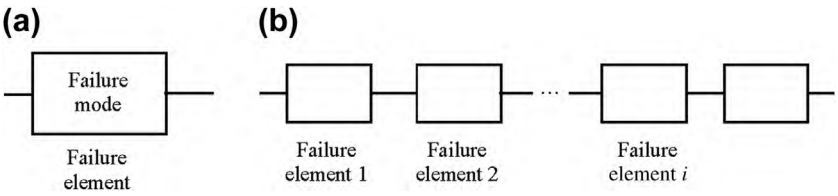


FIGURE 10.18

Block diagram for system reliability: (a) failure element and (b) series system.

10.5.1 SERIES SYSTEM

We first define a failure element to be a mathematic model of a specific failure mode at a specific location in the structure or a specific failure mode for a component in a mechanical system. A failure element is presented as a rectangular block in Figure 10.18(a).

One thing must be clarified first, and that is that "system" reliability does not necessarily mean that we are dealing with a mechanical system. This term is also used when there are multiple failure elements for a single component.

A series system, as shown in Figure 10.18(b), is one that fails when any failure element fails. For example, a frame structure with three members, shown in Figure 10.19, is considered. Each member is assumed to have two failure modes: stress yielding and buckling. As seen in this structure, load-carrying capacity is lost after any failure occurs in any individual component. A series system is also called a weakest-link system. Note that in the system shown, there are a total of six failure elements: two (yielding and buckling) for each member.

If the reliability of an individual failure element R_i is calculated, the reliability of a series system with m failure elements can be modeled as

$$R_{ss} = \prod_{i=1}^{m} R_i \tag{10.111}$$

For example, if a product has 20 failure elements ($m = 20$), each with a reliability of $R_i = 0.99$, the system reliability of the product is $R_{ss} = \prod_{i=1}^{20} R_i = (0.99)^{20} = 0.818$.

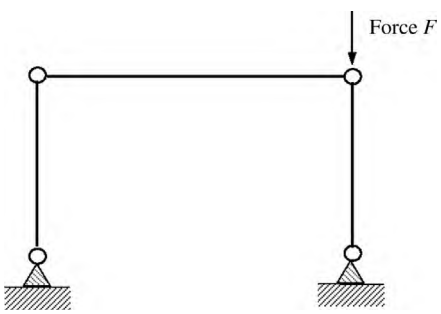


FIGURE 10.19

Frame structure with three truss members.

Note that, since reliability is less than or equal to 1, series system reliability is always less than that of the individual failure elements. Most consumer products exhibit series reliability.

10.5.2 PARALLEL SYSTEM

A much better arrangement is a parallel system, in which it is necessary for all failure elements in the system to fail for the system to fail. Such a system is represented in the block diagram shown in Figure 10.20. A system in which the components are arranged to give parallel reliability is said to be redundant; that is, more than one mechanism is necessary for the system functions to be carried out. In case one fails, a backup mechanism takes over to perform an identical function. In a system with full active redundancy, all but one component may fail before the system fails. For example, the frame structure with four members, shown in Figure 10.21, is holding a downward vertical load. Each member is assumed to have one failure mode, stress yielding. There are a total of four failure elements in this system. As seen in the structure, the system does not lose its load-carrying capacity unless all members fail.

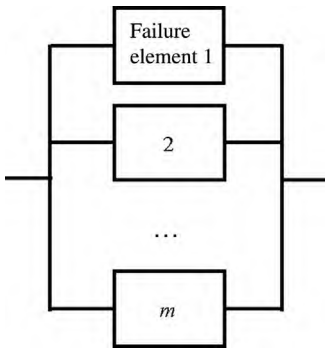


FIGURE 10.20

Reliability diagram for a parallel system.

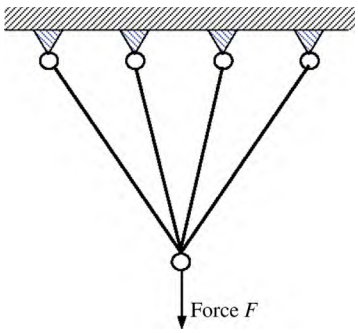


FIGURE 10.21

Frame structure with four members.

If the reliability of individual failure element R_i is available, the reliability of a parallel system with m failure elements can be modeled as

$$R_{ps} = 1 - \prod_{i=1}^{m} (1 - R_i) \tag{10.112}$$

For example, if a product has four failure elements ($m = 4$), each with a reliability of $R_i = 0.90$, then the system reliability of the product is

$$R_{ps} = 1 - \prod_{i=1}^{m} (1 - R_i) = 1 - (1 - 0.90)^4 = 0.9999$$

Note that reliability of a parallel system is always greater than that of individual failure elements. Although a parallel system offers better reliability, it is much more expensive to build because of its redundancy.

Some systems have partial active redundancy, in which certain components can fail without causing system failure but more than one component must remain functioning to keep the system operating. One example is a four-engine airplane, which can fly on two engines, but loses stability and control if only one engine is operating. This type of system is known as an n-out-of-m unit network. At least n units must function normally for the system to succeed rather than only one unit in the parallel case and all units in the series case.

The reliability of an n-out-of-m unit system is given by a binomial distribution:

$$R_{n|m} = \sum_{i=n}^{m} \binom{m}{i} R^i (1 - R)^{m-i} \tag{10.113}$$

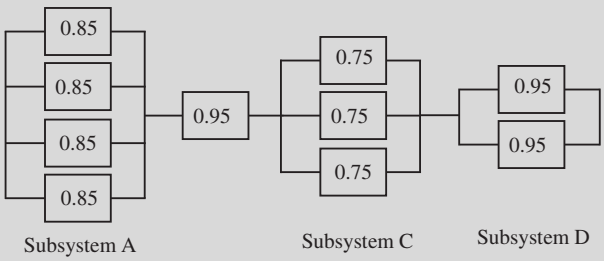
where

$$\binom{m}{i} = \frac{m!}{i!(m-i)!} \tag{10.114}$$

assuming the reliability of each failure element is identical and equal to R.

EXAMPLE 10.14

A complex engineering design is described by the reliability block diagram shown below. In subsystem A three components must operate for the subsystem to function successfully. In subsystem C, two components must operate for the subsystem to function. Subsystem D has true parallel reliability. We want to calculate the reliability of each subsystem and the overall system reliability.



EXAMPLE 10.14—cont'd

Solution

For subsystem A, the reliability R_A is

$$\begin{aligned} R_A = R_{3|4} &= \sum_{i=3}^{4} \binom{4}{i} 0.85^i (1-0.85)^{4-i} \\ &= \binom{4}{3} 0.85^3 (1-0.85)^{4-3} + \binom{4}{4} 0.85^4 (1-0.85)^{4-4} \\ &= 4 \times 0.85^3 (1-0.85)^1 + 1 \times 0.85^4 \times 1 = 0.8905 \end{aligned}$$

For subsystem C, the reliability R_C is

$$\begin{aligned} R_C = R_{2|3} &= \sum_{i=2}^{3} \binom{3}{i} 0.75^i (1-0.75)^{3-i} \\ &= \binom{3}{2} 0.75^2 (1-0.75)^{3-2} + \binom{3}{3} 0.75^3 (1-0.75)^{3-3} \\ &= 3 \times 0.75^2 (1-0.75)^1 + 1 \times 0.75^3 \times 1 = 0.8438 \end{aligned}$$

For subsystem D, the reliability R_D is

$$R_D = 1 - \prod_{i=1}^{2} (1 - R_i) = 1 - (1-0.95)^2 = 0.9975$$

The overall system reliability is then

$$R = R_{ss} = \prod_{i=1}^{4} R_i = R_A R_B R_C R_D = 0.8905 \times 0.95 \times 0.8438 \times 0.9975 = 0.7120$$

10.5.3 FORM APPROXIMATION FOR A SERIES SYSTEM

We assume that the reliability of individual failure elements in a system is known. In many applications, the reliability of individual failure elements is estimated using, for example, FORM. This subsection introduces FORM failure probability estimation for a series system. Instead of a thorough discussion here, we present important concepts and basic equations sufficient for an idea about this subject. For a detailed discussion with analytical examples, the reader may refer to Lee (2012) and Enevoldsen and Sørensen (1994).

As was shown earlier (Eq. 10.66), failure probability for a single failure element can be estimated using FORM as

$$P_f \approx P\big(L(\boldsymbol{u}) \le 0\big) = P\big(\beta - \boldsymbol{a}^{\mathrm{T}} \cdot \boldsymbol{u} \le 0\big) \tag{10.115}$$

in which the failure zone is approximated as $L(\boldsymbol{u}) \le 0$, and $\boldsymbol{a}$ is the normalized gradient of the limit state function at the MPP, which is also the gradient of the tangent line that passes the MPP.

For a series system with m failure elements, the failure probability for the ith element can be written as

$$P_f = P\big(g_{u_i}(\boldsymbol{u}) \le 0\big) \approx P\big(L_i(\boldsymbol{u}) \le 0\big) = P\Big(\beta_i - \boldsymbol{a}_i^{\mathrm{T}} \cdot \boldsymbol{u} \le 0\Big) = \Phi(-\beta_i) \tag{10.116}$$

where $g_{u_i}(\boldsymbol{u})$ and $L_i(\boldsymbol{u})$ are, respectively, the limit state function and its linearized Taylor expansion at the MPP for the ith failure element of the series system. $\boldsymbol{a}_i$ is the normalized gradient of the limit state function at the MPP for the ith failure element.

As discussed in Section 10.5.1, a series system fails if any one of the failure elements fails. Therefore, the failure region for a series system is the union of those regions of the individual failure elements, as illustrated in Figure 10.22(a), in which three limit state functions representing the respective failure elements are included in a two-dimensional problem. Using FORM, the failure region is first approximated by the union of linearized limit state functions at their respective MPPs, as shown in Figure 10.22(b).

The failure probability of the series system can be estimated as

$$\begin{aligned} P_f^s &= P\Big(\cup_{i=1}^m \big(g_{u_i}(\boldsymbol{u}) \le 0\big)\Big) \approx P\big(\cup_{i=1}^m \big(L_i(\boldsymbol{u}) \le 0\big)\big) \\ &= P\big(\cup_{i=1}^m \big(\beta_i - \boldsymbol{a}_i^{\mathrm{T}} \cdot \boldsymbol{u} \le 0\big)\big) = 1 - P\big(\cap_{i=1}^m \big(\beta_i - \boldsymbol{a}_i^{\mathrm{T}} \cdot \boldsymbol{u} \ge 0\big)\big) \end{aligned} \tag{10.117}$$

Note that the last term on the right of Eq. 10.117 has been obtained by applying the well-known De Morgan's law to the set $\cup_{i=1}^m (L_i(\boldsymbol{u}) \le 0) = \cup_{i=1}^m (\beta_i - \boldsymbol{a}_i^{\mathrm{T}} \cdot \boldsymbol{u} \le 0)$. Eq. 10.117 can be further reduced to

$$\begin{aligned} P_f^s &\approx 1 - P\Big(\bigcap_{i=1}^m \big(\beta_i - \boldsymbol{a}_i^{\mathrm{T}} \cdot \boldsymbol{u} \le 0\big)\Big) = 1 - P\Big(\bigcap_{i=1}^m \big(\boldsymbol{a}_i^{\mathrm{T}} \cdot \boldsymbol{u} \le -\beta_i\big)\Big) \\ &= 1 - \Phi_m(\beta, \boldsymbol{\rho}) \end{aligned} \tag{10.118}$$

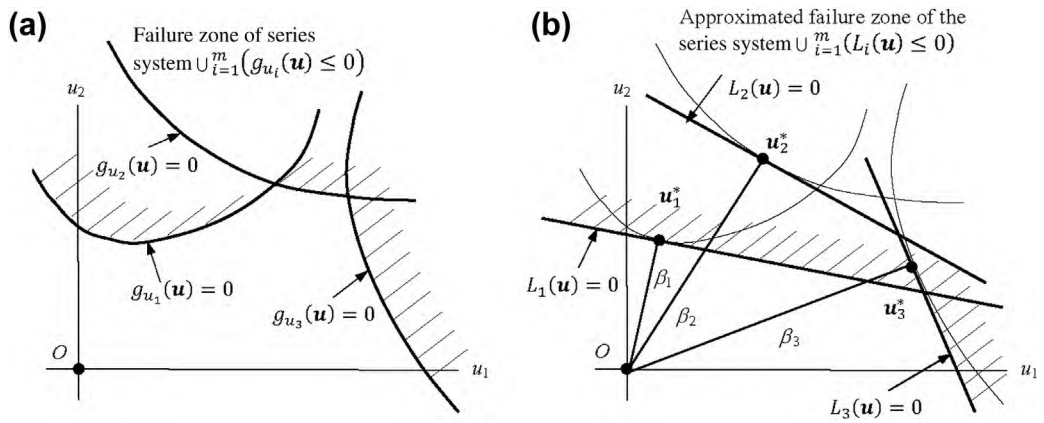


FIGURE 10.22

Failure probability estimate using FORM for the series system: (a) safe zone of the series system formed by $\cup_{i=1}^m (g_{u_i}(\boldsymbol{u}) \le 0)$; (b) safe zone of the series system formed by $\cup_{i=1}^m (L_i(\boldsymbol{u}) \le 0)$.

in which Φ_m is the m-dimensional normal distribution function defined as

$$\Phi_m(\beta, \boldsymbol{\rho}) \equiv \int_{-\infty}^{\beta_1} \int_{-\infty}^{\beta_2} \cdots \int_{-\infty}^{\beta_m} \boldsymbol{\phi}_m(\boldsymbol{x}, \boldsymbol{\rho}) dx_1 dx_2 \ldots dx_n \tag{10.119}$$

where $\boldsymbol{\phi}_m(\boldsymbol{x},\boldsymbol{\rho})$ is the m-dimensional normal probability density function defined as

$$\boldsymbol{\phi}_m(\boldsymbol{x}, \boldsymbol{\rho}) = \frac{1}{(2\pi)^{m/2}\sqrt{|\boldsymbol{\rho}|}} e^{-\frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\rho}^{-1}\boldsymbol{x}} \tag{10.120}$$

In Eqs 10.118–10.120, $\boldsymbol{\rho}$ is an $m \times m$ matrix, defined as

$$\rho_{ij} = \boldsymbol{a}_i^{\mathrm{T}} \cdot \boldsymbol{a}_j \tag{10.121}$$

It can be shown that ρ_{ij} is the correlated coefficient of random variables X_i and X_j (Sørensen, 2004). $|\boldsymbol{\rho}|$ in Eq. 10.120 is the determinant of the Jacobian of the matrix $\boldsymbol{\rho}$.

The descriptions just given show that it is very important to be able to calculate the multidimensional standard normal distribution function $\boldsymbol{\phi}_m(\boldsymbol{x}, \boldsymbol{\rho})$, which, except for special cases, must be estimated by approximation. We simply mention a few popular methods, in decreasing order of general precision:

- Gollwitzer and Rackwitz's approximation (Gollwitzer and Rackwitz, 1983)
- Hohenbichler's approximation (Hohenbichler, 1984)
- Average correlation coefficient approximation (Thoft-Christensen and Sørensen, 1982)
- Ditlevsen's bounds (Ditlevsen, 1979) for series systems
- Simple bounds (Cornell, 1967).

For a series system in which all failure modes are fully correlated, it is realized that the failure probability is equal to the failure probability of the failure element with the largest failure probability—in this case, a system where the weakest link may be clearly identified. As the correlation between the failure modes is somewhere between zero and one, the simple bounds on the failure probability of a series system may thus be given as

$$\max_{i=1}^{m}\{P(g_{u_i}(\boldsymbol{u}) \le 0)\} \le P_f^s \le \sum_{i=1}^{m}\left\{P\left(g_{u_i}(\boldsymbol{u}) \le 0\right)\right\} \tag{10.122}$$

where the lower bound corresponds to the exact value of P_f^s if all the elements in the series system are fully correlated.

In terms of reliability indices, Eq. 10.122 can be written as

$$-\Phi^{-1}\left(\sum_{i=1}^{m}\Phi^{-1}(-\beta_i)\right) \le \beta^s \le \min_{i=1}^{m}\beta_i \tag{10.123}$$

When the failure of one failure element does not dominate in relation to the other failure elements, the simple bounds are generally too wide and therefore often of minor interest for practical use.

10.6 GENERAL-PURPOSE RELIABILITY ANALYSIS TOOLS

As discussed in Section 10.4.1, a limit state function associated with the respective failure mode must be defined before failure probability can be calculated. Although in this chapter we have included limit state functions of structural problems, the methods and theory presented are not restricted to structural problems. For example, if we are dealing with a rigid-body dynamic problem (as discussed in Chapter 8) in the design of a mechanical system, there may be a need to calculate the failure probability of a critical performance measure identified in the system because of uncertainties in parameters, such as mass properties or size dimensions of individual moving components, external forces, and the like.

One instance might be the design of a single-piston engine (see Chapter 8) in which the maximum dynamic load applied to the connecting rod resulting from the firing load and the inertia of the individual components must not be greater than an upper limit because of concern about the rod's structural integrity. Uncertainties in physical parameters may have to be considered, such as magnitude of the firing force, geometric dimension variation in various components including the rod, and mass density of the rod, which can be characterized by certain distributions determined by physical measurement. The failure probability of the limit state function associated with the load failure mode can be estimated using reliability analysis methods, such as Monte Carlo simulation or FORM, as discussed.

Thus far, limit state functions have been represented by a simple algebraic function of random variables representing the aforementioned uncertainties. In these examples, we can easily perform reliability analysis. In dealing with many complex structural systems or failure mechanisms, however, evaluation of limit state functions requires sophisticated structural analyses or mechanism simulations employing analysis tools.

Although significant progress has been made in reliability analysis methods, the development of corresponding software is still lagging behind (Pellissetti and Schuëller, 2006), especially in the commercial sector. There are very few options in commercial reliability analysis tools. Those available have been developed mainly to support reliability analysis for structural problems that require finite element analysis for evaluating limit state functions—often termed finite element-reliability analysis (FE-RA) (e.g., Sudret and Der Kiureghian, 2000; Lee et al., 2008). In this section, we briefly review some FE-RA software tools that are readily (i.e., relatively) available for use. Note that many of the existing FE-RA software packages are primarily for component reliability analysis, in which the FE reliability analysis is performed for individual failure modes of a structural member or location that are represented by single limit state function.

One of the most reputable reliability analysis software programs is NESSUS® (www.nessus.swri.org), a modular computer software program for performing probabilistic analysis of structural/mechanical components and systems. NESSUS was initially developed by the Southwest Research Institute (SwRI) for NASA to perform probabilistic analysis of space shuttle main engine components. SwRI continues to develop and apply NESSUS to a diverse range of problems, among them aerospace structures, automotive structures, and biomechanics. Instead of including its own FE code, NESSUS has been interfaced with many well-known third-party and commercial deterministic analysis programs, including ABAQUS®, ANSYS®, MSC/Nastran®, and MATLAB.

Integrating with commercial third-party FE software tools seems to be the most logical and economical approach for structural reliability analysis. The only drawback is that the type of limit state functions are restricted to those offered by the FE tools. However, these tools are powerful and support

a broad range of structural analysis problems. Even if certain types of structural problem or finite element are not found in one code, integrating with multiple FE software tools greatly diminishes this concern. Moreover, the same approach supports other types of engineering problems when proper analysis software is integrated to provide results for the limit state functions. Details can be found in Schuëller and Pellissetti (2008).

There are several other commercial tools that follow the same approach, such as COSSAN (www.cossan.co.uk), Pro-FES® (www.ara.com), STRUREL (www.strurel.de/feat.htm), Proban (www.dnvusa.com), FReET (www.cervenka.cz), and UNIPASS® (www.predictionprobe.com).

Many research or semi-research codes are worthy of mention here. These codes, developed by research groups in universities, are occasionally upgraded; they are offered as free downloads in most cases and are customizable for various purposes, such as graduate research. One code in this category is FERUM (Finite Element Reliability Using MATLAB, http://www.ifma.fr/lang/es/Recherche/Labos/FERUM), which is primarily intended for pedagogical purposes although it is also useful for research and engineering production. FERUM is a set of functions within MATLAB that carry out finite element-reliability analysis as well as reliability analysis for prescribed analytical limit state functions. Another code in this category is CalREL (www.ce.berkeley.edu), a general-purpose structural reliability analysis program that is available for purchase from UC Berkeley in both object and source codes.

All of the noncommercial software codes mentioned offer excellent reliability analysis capabilities, including FORM, SORM, Monte Carlo simulation, and response surface. However, the ANSYS Probabilistic Design System (PDS) is the only commercial FE tool with reliability analysis capability. ANSYS PDS automates the reliability analysis process. Using simple menu picks (or commands), users can specify many input variables and their variations in statistical terms (Gaussian, Weibull, etc.). ANSYS then manages the many runs that are necessary for an accurate reliability estimate. Instead of more advanced FORM or SORM, ANSYS PDS offers Monte Carlo simulation and the response surface method.

10.7 CASE STUDY

A tracked vehicle roadarm is presented here as a case study to demonstrate reliability analysis in practical failure probability calculation. A deterministic fatigue life prediction of the roadarm was discussed in Section 9.8.1 together with a rigid-body dynamic simulation of the tracked vehicle on the Aberdeen Proving Ground 4 (APG4). Here random variables are defined and probabilistic fatigue life predictions are made for the roadarm.

As discussed in Section 9.8.1, a rigid-body dynamic simulation on the APG4 was carried out first to obtain loads applied to the roadarm for a total of 20 seconds in simulation. A finite element analysis was performed to obtain the roadarm's stress influence coefficients (SICs), using ANSYS to apply 18 quasistatic loads. The dynamic stresses at finite element nodes were then calculated by superposing the SICs with their corresponding external forces and accelerations and velocities in the time domain obtained from the dynamic simulation. To compute the multiaxial crack initiation life of the roadarm, the equivalent von Mises strain approach (see Chapter 9) was employed. The fatigue life contour is given in Figure 10.23. The shortest fatigue life of the roadarm was identified at node 1216 (Chang et al., 1997).

The random variables and their statistical values for the crack initiation life prediction are listed in Table 10.5, including those for material and tolerance. The eight tolerance random variables were

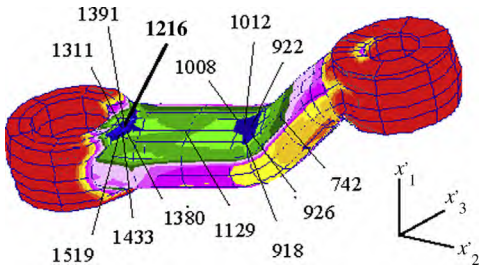


FIGURE 10.23

Crack initiation life contour for the tracked vehicle roadarm.

Table 10.5 Random Variables for Crack Initiation Life Prediction

Random Variables	Mean Value	Standard Deviation	Distribution
Young's modulus, E	30.0E+6	0.75E+6	Lognormal
Fatigue strength coefficient, σ_f' (psi)	1.77E+5	0.885E+4	Lognormal
Fatigue ductility coefficient, ϵ_f'	0.41	0.0205	Lognormal
Fatigue strength exponent, b	−0.07300	0.00365	Normal
Fatigue ductility exponent, c	−0.6	0.003	Normal
Tolerance b1 (in.)	3.2496	0.032450	Normal
Tolerance b2 (in.)	1.9675	0.019675	Normal
Tolerance b3 (in.)	3.1703	0.031703	Normal
Tolerance b4 (in.)	1.9675	0.019675	Normal
Tolerance b5 (in.)	3.1703	0.031703	Normal
Tolerance b6 (in.)	2.6352	0.026352	Normal
Tolerance b7 (in.)	3.2496	0.032496	Normal
Tolerance b8 (in.)	5.0568	0.050568	Normal

defined to characterize the four cross-sectional shapes of the roadarm, as shown in Figure 10.24(a) (Yu, et al., 1997). The profile of the cross-sectional shape was composed of four straight lines and four cubic curves (Figure 10.24(b)). Side variations (x'_1-direction) of the cross-sectional shapes are defined as random variables b1, b3, b5, and b7 for sections 1 through 4, respectively (Figure 10.24(b)). The vertical variations (x'_3-direction) of the cross-sectional shapes are defined as the remaining four random variables (Figure 10.24(b)).

The first-order reliability method (FORM) was used to calculate the failure probability of the crack initiation life. The deterministic fatigue life at node 1216 was the shortest, with 9.63E + 06 blocks (20 sec per block). The cumulative distribution function of the crack initiation life (number of blocks to failure) at node 1216 is shown in Figure 10.25. The horizontal axis in the figure is the required number of service blocks, and the vertical axis is the failure probability. The CDF in the figure was obtained by

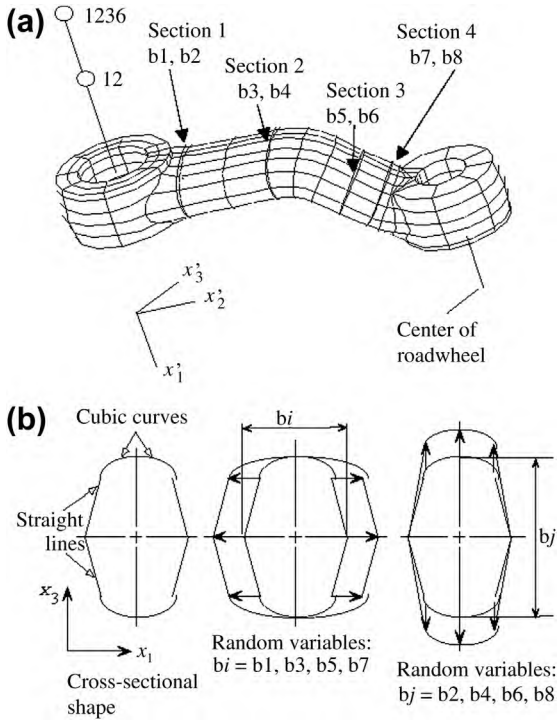


FIGURE 10.24

The tracked vehicle roadarm: (a) FE model with four sections and (b) geometric parameters included for modeling uncertainty in manufacturing tolerance.

reliability analysis at the seven required numbers of service blocks, ranging from 1×10^5 to 5×10^6 blocks, which are marked in the figure.

As discussed previously, one FORM analysis is equivalent to solving a deterministic optimization. For the roadarm example, seven FORM analyses were performed to create the graph shown in Figure 10.25. In actual design applications, the CDF curve can be used to obtain the failure probability for the required number of service blocks before crack initiation, or the required number of service blocks before crack initiation with a required failure probability. For example, it can be seen in the figure that, if the required number of service blocks before crack initiation is 3.0E + 06, the failure probability is about 11%.

10.8 SUMMARY

In this chapter, we demonstrated that a deterministic approach using the safety factor method or the worst-case scenario is not sufficient to provide a full picture of the safety or reliability of a product design. These methods can lead to a design that is either not as reliable as desired or to a product or component that is overdesigned. A safe and reliable product can be ensured only by bringing probabilistic or statistical considerations into the design process.

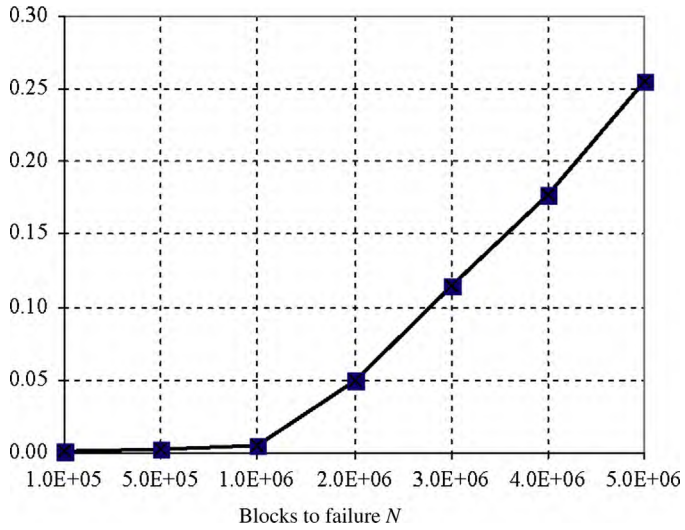


FIGURE 10.25

CDF graph of failure probability for the roadarm crack initiation life.

Mechanical engineers must understand the importance of the probabilistic aspect of product design, and they must be able to perform adequate reliability analysis in solving various engineering problems. For this reason, we introduced several basic but widely used methods for reliability analysis: FORM, SORM, Monte Carlo simulation, importance sampling, and response surface. None of these is a clear-cut choice above the others. All have strengths and weaknesses. The key is to understand how each works and to choose the one appropriate for solving the problem at hand—that is, the one that requires the least computational effort to achieve an accurate enough estimate of failure probability. For general applications in engineering design, evaluation of limit state functions can require substantial computation, such as analyzing large-scale structures using finite element method.

As mentioned in this chapter, one way to address such applications is to search the MPP and use FORM to estimate failure probability, and then use MPP-based importance sampling to further improve the accuracy of the FORM estimate if necessary. In using FEA for evaluating the limit state function, the designer may develop an interface program to integrate an FEA code into reliability analysis code. Details are provided in Schuëller and Pellissetti (2008). Alternatively, the FEA input data file may be manually modified in accordance with the random variable values.

In either case, it is important for design engineers to understand the importance of reliability in product design from a probabilistic perspective and to be able to use adequate methods to obtain failure probability estimates. With this in mind, design engineers should incorporate failure probability into their formulation of design problems and, for accurate estimates of product failure probability, strive to acquire information and statistical data to characterize physical parameters involved in the limit state functions.

In this chapter, we focused on reliability analysis and only touched a bit on design. More in-depth discussion of engineering design for reliability, including reliability-based design optimization, is provided in Chapter 19.

QUESTIONS AND EXERCISES

10.1. Let $\boldsymbol{X} = [X_1, X_2 \ldots, X_n]^{\mathrm{T}}$ be a vector of random variables with normal distribution, and let $\boldsymbol{Y} = [Y_1, Y_2 \ldots, Y_n]^{\mathrm{T}}$ be the corresponding vector of random variables of standard normal distribution, which is obtained by transforming random vector X_i, for $i = 1, n$, by

$$y_i = \frac{x_i - \mu_{x_i}}{\sigma_{x_i}}$$

Show that the covariance matrix of $\boldsymbol{Y}$ (that is, $\boldsymbol{C}_Y$, as defined in Eq. 10.83) is equal to the correlation coefficient matrix of $\boldsymbol{X}$ (that is, $\boldsymbol{\rho}_X$, as defined in Eq. 10.88):

$$\boldsymbol{C}_Y = \boldsymbol{\rho}_X$$

10.2. Let $\boldsymbol{Y} = [Y_1, Y_2 \ldots, Y_n]^{\mathrm{T}}$ be a vector of random variables of the standard normal distribution transforms with covariance matrix $\boldsymbol{C}_Y$. Show that a matrix $\boldsymbol{T}$ defined as shown transforms random variables $\boldsymbol{Y}$ of correlated standard normal distribution into $\boldsymbol{U} = [U_1, U_2 \ldots, U_n]^{\mathrm{T}}$ that are random variables of uncorrelated (independent) standard normal distribution:

$$\boldsymbol{Y} = \boldsymbol{T}\boldsymbol{U}$$

where

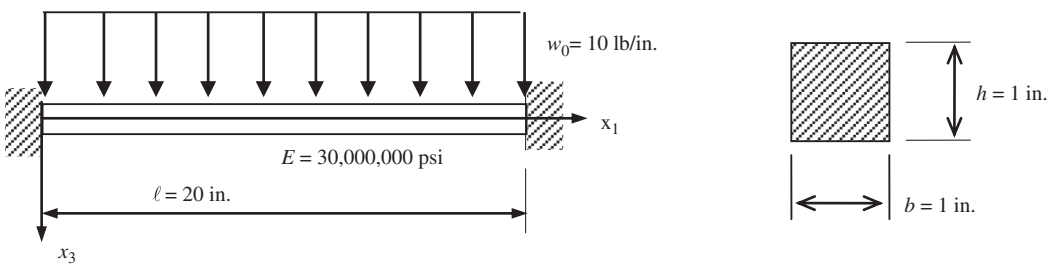
$$T_{jj} = \sqrt{1 - \sum_{k=1}^{j-1} T_{jk}^2},\ j = 1, n$$

and

$$T_{ij} = \frac{\rho_{ji} - \sum_{k=1}^{j-1} T_{jk} T_{ik}}{T_{jj}},\ i > j$$

10.3. Derive Eq. 10.109 from Eq. 10.108.

10.4. A beam clamped at both ends, shown in the following figure, is subject to reliability assessment for its current design.



Assume a displacement failure mode that states that the maximum displacement of the beam cannot exceed 0.02 in. If two random variables, the width and height of the cross-sectional area, are considered, calculate the failure probability using Monte Carlo simulation. Note that both random variables are uncorrelated and of normal distribution, with their respective mean values and standard deviations defined as $X_b \sim N(1, 0.1)$ in. and $X_h \sim \mathrm{N}(1, 0.15)$ in.

10.5. Continue with Problem 4, but use FORM to estimate failure probability. Calculate the MPP and find the reliability index β. Compare the failure probability found using FORM with that found by Monte Carlo simulation. Is the FORM result accurate compared with the Monte Carlo result?

10.6. Use the MPP obtained from Problem 5 to estimate the failure probability of the same beam problem using MPP-based importance sampling. How many sample points are needed to achieve an accurate failure probability?

10.7. If the width and height random variables of the beam are correlated with correlated coefficient $\rho_{bh} = 0.25$, transform them into a U-space of standard normal distribution and express the limit state function in terms of random variables u_b and u_h.

10.8. Use the response surface method to solve the same beam problem. Find a quadratic function $\widehat{g}(\boldsymbol{x})$ to approximate the true limit state function $g(\boldsymbol{x})$. Use the following two sets of sample points:

Set A: [–1,1], [0,1], [1,1], [–1,0], [0,0], [1,0], [–1,–1], [0,–1], [1,–1]
Set B: [–2,1], [–1,1], [0,1], [1,1], [2,1], [–2,0], [–1, 0], [0,0], [1,0], [2,0], [–2,–1], [–1,–1], [0,–1], [1,–1], [2,–1]

Compare the functions $\widehat{g}_A(\boldsymbol{x})$ and $\widehat{g}_B(\boldsymbol{x})$ obtained from these respective sample data sets. Use Monte Carlo simulation to calculate failure probabilities using both functions $\widehat{g}_A(\boldsymbol{x})$ and $\widehat{g}_B(\boldsymbol{x})$. Which one gives a more accurate failure probability estimate? Can you explain why one function is better than the other?

REFERENCES

Boggs, P.T., Tolle, J.W., 1995. Sequential quadratic programming. Acta Numerica 4, 1–51.

Chang, K.H., Yu, X., Choi, K.K., 1997. Shape Design Sensitivity Analysis and Optimization For Structural Durability. International Journal of Numerical Methods in Engineering 40, 1719–1743.

Choi, S.K., Grandhi, R.V., Canfield, R.A., 2006. Reliability-based Structural Design. ISBN-10: 1-84628-444-9. Springer.

Cornell, C.A., 1967. Bounds on the reliability of structural systems. ASCE Structural Division Journal 93 (l), 171–200.

Ditlevsen, O., 1979. Narrow reliability bounds for structural systems. Journal of Structural Mechanics 7 (4), 453–472.

Ditlevsen, O., Madsen, H.O., 1996. Structural Reliability Methods. John Wiley & Sons.

Enevoldsen, I., Sørensen, J.D., 1994. Reliability-based optimization in structural engineering. Structural Safety 15, 169–196.

Gollwitzer, S., Rackwitz, R., 1983. Equivalent components in first-order systems reliability. Reliability Engineering 5, 99–115.

Hasofer, A.M., Lind, N.C., 1974. An Exact and Invariant First Order Reliability Format. Journal of the Engineering Mechanics Division, ASCE Vol. 100, 111–121.

Hohenbichler, M., 1984. An asymptotic formula for the probability of intersections, Berichte zur Zuverltissigkeitstheorie der Bauwerke. Heft 69, LKI. Technische Universitlt München.

Hurtado, J.E., 2004. Structural Reliability: Statistical Learning Perspectives. Springer-Verlag.

Lee, Y.J., 2012. Finite-element-based system reliability analysis and updating of fatigue-induced sequential failures. PhD dissertation. University of Illinois at Urbana-Champaign.

Lee, Y.-J., Song, J., Tuegel, E.J., 2008. Finite element system reliability analysis of a wing torque box. In: Proceedings 10th AIAA Nondeterministic Approaches Conference April 7–10.

Lee, J.-O., Yang, Y.-S., Ruy, W.,-S., 2002. A comparative study on reliability-index and target-performance-based probabilistic structural design optimization. Computers and Structures 80, 257–269.

Liu, P.L., Der Kiureghian, A., 1986. Multivariate distribution models with prescribed marginals and covariances. Probabilistic Engineering Mechanics 1 (2), 105–112.

Nataf, A., 1962. Détermination des distribution dont les marges sont données. Comptes rendus de l'academie des sciences 225.

Pellissetti, M.F., Schuëller, G.I., 2006. On general purpose software in structural reliability. Structural Safety 28 (1–2), 3–16.

Rackwitz, R., Fiessler, B., 1978. Structural Reliability Under Combined Random Load Sequences. Journal of Composite Structures 9, 489–494.

Rosenblatt, M., 1952. Remarks on a multivariate transformation. Annals of Mathematical Statistic 23, 470–472.

Schuëller, G.I., Pellissetti, M.F., 2008. Computational tools for structural analysis with uncertainties: Software technology and large-scale applications. In: Topping, B.H.V., Papadrakakis, M. (Eds.), Trends in Computational Structures Technology. Saxe-Coburg Publications.

Sørensen, J.D., 2004. Notes in Structural Reliability Theory and Risk Analysis. Aalborg University, Aalborg.

Sudret, B., Der Kiureghian, A., 2000. Stochastic finite element methods and reliability: A state-of-the-art report. Department of Civil and Environmental Engineering, University of California, Berkeley.

Thoft-Christensen, P., Sørensen, J.D., 1982. Reliability of structural systems with correlated elements. Applied Mathematical Modelling 6, 171–178.

Wei, D., 2006. A univariate decomposition method for higher order reliability analysis and design optimization. PhD dissertation. University of Iowa.

Yu, X., Choi, K.K., Chang, K.H., 1997. A Mixed Design Approach for Probabilistic Structural Durability. Journal of Structural Optimization 14, 81–90.

CHAPTER

VIRTUAL MACHINING 11

CHAPTER OUTLINE

11.1 Introduction 601
11.2 NC Part Programming 602
11.2.1 Basics of NC Machines 602
11.2.2 Basic Concept of Part Programming 606
11.2.3 Computer-Assisted Part Programming 608
11.2.4 CAD/CAM Approach 611
11.3 Virtual Machining Simulations 611
11.3.1 Basic Machining Simulations 612
11.3.2 Advanced Machining Simulations 616
11.3.3 Turning Simulations 621
11.4 Practical Aspects in CNC Machining 622
11.4.1 Jigs and Fixtures 623
11.4.2 Cutters and Machining Parameters 624
11.4.3 Setting a CNC Sequence 627
11.5 Commercial Machining Simulation Software 629
11.5.1 General-Purpose Machining Software 629
11.5.2 Special-Purpose Machining Software 630
11.6 Case Study and Tutorial Examples 632
11.6.1 Case Study 632
11.6.1.1 Virtual Machining for Green Part 635
11.6.2 Tutorial Examples 636
11.6.2.1 Name Plate 636
11.6.2.2 Block with a Sculpture Surface 637
11.7 Summary 640
Appendix 11A: Sample Address Codes 641
Appendix 11B: Sample G- and M-Codes 642
Questions and Exercises 643
References 646

In product development, it is highly desirable that the manufacturability of the individual parts and the overall assembly be verified prior to functional prototyping and production. Discovering manufacturing-related issues during the late stages causes delay in bringing the product to market and increases product development cost. It is essential to address manufacturability issues during the

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00011-9
Copyright © 2015 Elsevier Inc. All rights reserved.

product design stage, and it is effective to address such issues using virtual manufacturing technology and software tools.

Virtual manufacturing is the use of simulation-based technology to aid engineers in defining, simulating, and visualizing the manufacturing process of a product in a computer environment. By using virtual manufacturing, the manufacturing process can be defined and verified early in the design stage. Some, if not all, of the potential manufacturing-related issues can be detected and addressed while the design is still being finalized. In addition, manufacturing cost and time, which constitute a significant portion of the product cost, can be estimated.

Virtual manufacturing is a broad subject that involves a substantial range of topics. Instead of addressing broad aspects of manufacturing technology and process, we will focus, in this chapter, on the machining operations of virtual manufacturing, such as milling, turning, and drilling; namely, virtual machining (VM). In addition, sheet forming simulation, which has recently gained more popularity in industry, will be discussed in Chapter 13.

Virtual machining allows designers to conduct machining process planning, generate machining toolpaths, visualize and simulate machining operations, and estimate machining time. Moreover, the toolpath generated can be converted into CNC codes (M-codes and G-codes) to machine functional parts as well as die or mold for production. One of the main considerations in discussing virtual machining in this chapter is that CNC machining has a relatively low setup cost compared to the forming, molding, or casting process. In general, CNC machines are relatively accessible to small businesses and academia. Furthermore, CNC is cost effective for low-volume production, which is ideal in support of physical or functional prototyping.

In most cases, the toolpath is generated in a so-called CL (cutter location) data format and then converted to CNC codes using corresponding post-processors. In addition to basic virtual machining simulations, we will briefly discuss CNC codes, or part programming, to help you become familiar with and be able to read (and hopefully edit and write) CL data and CNC codes.

Although "Virtual Machining" is the title of this chapter, we will include a few practical aspects of using CNC mills for machining. We will discuss choosing essential machining parameters, such as feedrate and spindle speed, in addition to cutter and fixture selections.

Machining of dies that support die casting for tracked vehicle roadarms is included as an example to illustrate and demonstrate the application of VM to a practical engineering design. In addition, three practice examples—profile milling, volume milling, and surface milling—using both Pro/MFG (www.ptc.com/product/creo/machining-extension) and Mastercam (www.mastercam.com) will be discussed. Some of these practice examples can also be found in the tutorial lessons, which include CAMWorks (www.camworks.com) in addition to Pro/MFG and Mastercam. Detailed instructions for bringing up these models and steps for carrying out the VM discussed in this chapter can be found in Projects P4, M4, and S4, for Pro/MFG, Mastercam, and CAMWorks, respectively. Example models are available for download at the book's companion website (http://booksite.elsevier.com/9780123820389).

This chapter was written with the assumption that readers are familiar with basic manufacturing processes, especially machining. Details related to basic milling, turning, and hole making can be found in other textbooks, such as *Manufacturing, Engineering & Technology*, 6th ed., by Serope Kalpakjian and Steven R. Schmid (see references at the end of this chapter). In addition, we encourage readers to review excellent NC programming books (such as Smid, 2000) before going over this chapter.

The overall objectives of this chapter are to (a) provide you with a general understanding of virtual machining simulations and how to employ virtual machining software to support product design; (b) become familiar with existing commercial software for virtual machining; and (c) be able to use Pro/MFG, Mastercam, or CAMWorks for basic machining jobs (after going through the tutorial lessons).

11.1 INTRODUCTION

In the mechanical and aerospace industries, engineers often confront the challenge of designing components that are capable of sustaining structural loads and meeting functional requirements. It is imperative that these components contain minimum material to reduce cost and increase the efficiency of the mechanical system in terms of, for example, fuel consumption of a ground vehicle. The geometry of these components is usually complicated due to strength and efficiency requirements, which often results in increased manufacturing time and cost. Other significant issues (e.g., machinability, characterized by surface finish; tool (cutter) life; and force and power requirements) add to the challenges that designers must confront. In product design, it is essential that parts designed can be manufactured or machined with regular and off-the-shelf set ups and in the shortest time possible since machining time directly impacts product costs.

Various manufacturing processes and technologies are suitable for part manufacturing, including casting, forming, molding, sintering, and machining, which have been well documented (Kalpakjian and Schmid, 2010). Among them, machining usually has a relatively low setup cost compared to forming, molding, and casting processes. Consequently, CNC machines are relatively accessible to small businesses and academia. In addition, CNC is cost-effective for low-volume production, which is ideal for physical or functional prototyping.

In line with the main theme of Part III Product Manufacturing and Cost Estimating, this chapter focuses on virtual machining, which is the use of simulation-based technology to aid designers in conducting machining process planning in a virtual environment, generating machining toolpath, visualizing and simulating the machining operations, and estimating machining time.

Simulation of a mold machining is illustrated in Figure 11.1, where three NC sequences are defined. The first sequence creates a smooth flat surface on top of the workpiece; that is, face milling.

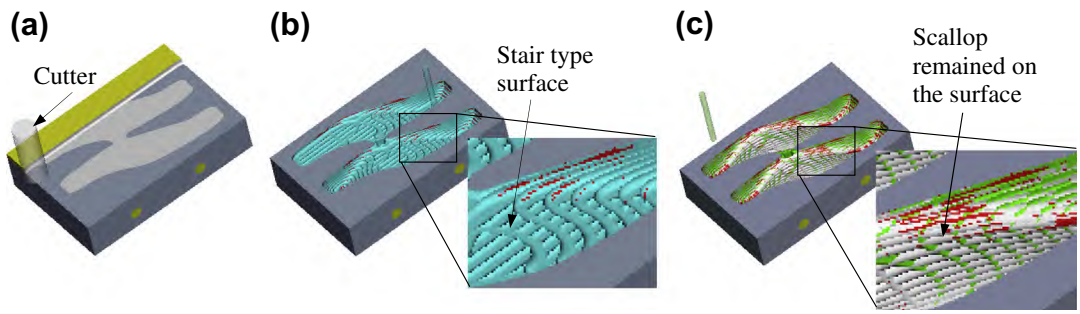


FIGURE 11.1

Virtual machining for a cover die, (a) face milling, (b) volume milling (rough cut), and (c) contour surface milling (finish cut).

The second sequence cuts the cavity (a rough cut) and generates a stair-type surface, which is not smooth. Note that the distance between the stair layers is controlled by a machining parameter, called step depth—set to be 0.1 in. in this case. The third sequence uses a contour surface milling to polish the cavity's surface (a finish cut). This minimizes the grinding operation and increases accuracy of the die.

As illustrated in Figure 11.1(c), after the three NC sequences the cavity surface is still not completely smooth—scallop and tool marks are clearly visible. The maximum scallop height is computed as less than 0.05 in. The size of the scallop remaining on the cavity surface can be reduced by using a smaller stepover or a smaller size cutter in the finish cut. Machining time will most likely increase using either way. The tool marks can be removed by keeping a small layer of the material on the bottom face of the cavity uncut during the rough cut. Usually, grinding operations are employed to further polish the cavity's surface.

In this chapter, we will start by a short and brief introduction on NC part programming, then followed by a few virtual machining examples. After reviewing these two topics, you should have a fairly complete picture of virtual machining. We will then discuss a few important points you should be aware of in CNC machining from a practical perspective. This will help you create valid CNC codes and take practical issues into consideration. We will also discuss commercially available machining simulation software tools, which should help you make an adequate selection suitable for your needs. This chapter wraps up by introducing a case study that involves virtual machining for a tracked vehicle roadarm. In addition, tutorial examples are provided.

11.2 NC PART PROGRAMMING

NC part programming creates NC codes, which provide the instructions that drive cutters and control machine operations. In general, there are three approaches supporting NC programming: manual, computer-assisted, and CAD/CAM. In this section, we will first briefly go over these approaches. We focus on the CAD/CAM approach in the remainder of the chapter. Before introducing NC part programming, we will provide a brief discussion on NC machines, coordinate systems, type of machines, and type of machining operation in NC.

11.2.1 BASICS OF NC MACHINES

The difference between NC machines and conventional machines is in the way in which the various functions and cutter movements are controlled. In conventional machines, these are controlled by shop mechanists. In NC, these motions are controlled by the machine control unit (MCU), as depicted in Figure 11.2.

The MCU (brain of the NC) consists of a DPU (data processing unit) and a CLU (control loop unit). DPU reads the part program from tape, or some other media, and decodes the part program statements, processes the decoded information, and passes information to the CLU. The information includes: position of each axis of the machine, its direction of motion, feedrate, and auxiliary function control signals (e.g., coolant on or off). CLU receives data from the DPU, converts them to control signals, and controls the machine via actuation devices that replace the hand wheel of the conventional machine. An actuation device could be a servomotor, a hydraulic actuator, or a step motor. A servomotor (or servo) is an electromechanical device in which an electrical input determines the position of the armature of a motor. Servos are used extensively in robotics and radio-controlled cars, airplanes, and boats.

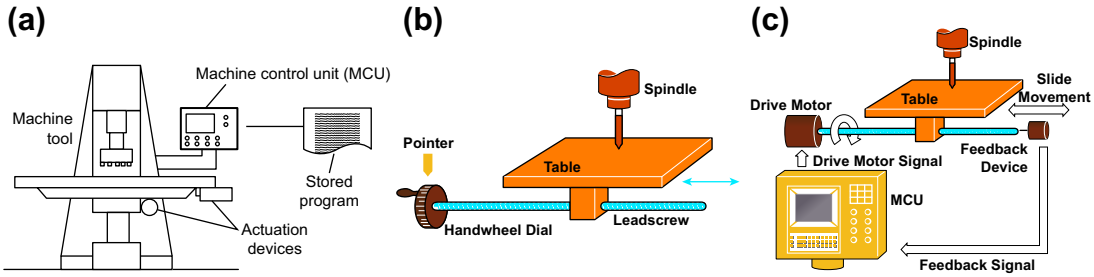


FIGURE 11.2

Configuration of a typical NC machine: (a) the machine control unit, (b) hand wheel dial, and (c) closed-loop control.

The MCU gives instructions to the servo system, monitors both the position and velocity output of the system, and uses this feedback to compensate for errors between the program command and the system response. The instructions given to the servos are modified according to the measured response of the system, called closed-loop control.

Each axis of motion is equipped with a driving (actuation) device. The primary three axes of motion are referred to as the *X*-, *Y*-, and *Z*-axes. They form the machine tool coordinate system. The *XYZ* system is a right-hand system and the location of its original may be fixed (older machine) or adjustable (floating zero).

The *Z*-axis is the most important axis for machining. This axis is always aligned with the spindle that imparts power, as shown in Figure 11.3. The spindle may rotate a workpiece such as in a lathe or it may rotate a tool as in a milling machine. Usually, the direction that moves away from the workpiece is defined as positive.

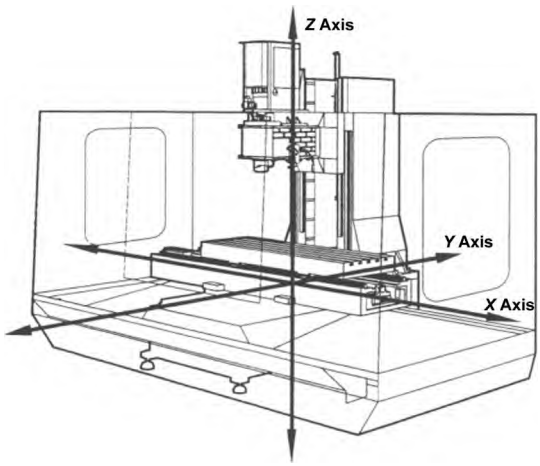


FIGURE 11.3

XYZ system: mill (HAAS).

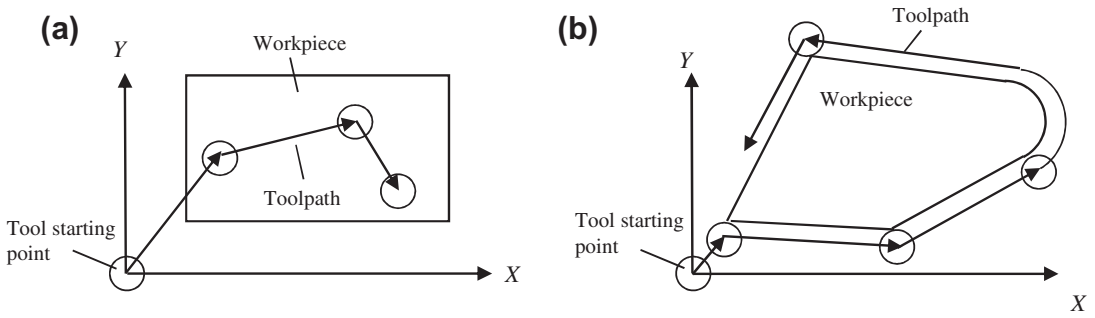


FIGURE 11.4

Types of NC machines: (a) point-to-point and (b) continuous path.

On a workpiece-rotating machine (e.g., lathe), the *X*-axis is the direction of tool movement, and a motion along its positive direction moves the tool away from the workpiece. On a milling or drilling machine, the positive *X*-axis points to the right when the programmer is facing the machine (Figure 11.3). Note that the definition of the positive *X*-axis is not universal. *Y*-axis is determined by *X*- and *Z*-axes through the right-hand rule. *W*-axis (less common) is parallel to *Z*-axis, but points in the opposite direction.

There are several ways to categorize NC machines. By looking at its cutting path, a NC machine can be a PTP (point-to-point) or continuous path (Figure 11.4). For a PTP machine, the cutter performs operations on the workpiece at specific locations. The cutter is not always in contact with the workpiece throughout its motion or its path. The exact path the tool takes in moving from point to point is in general immaterial (except that tool-traveling time must be minimized and not collide with the workpiece and fixtures). A hole-drilling machine is a good PTP example. On the other hand, for a continuous path NC, the cutter is mostly in contact with the workpiece during its motion or its path. The workpiece is being affected throughout the toolpath. The entire travel of the cutting tool must be controlled to close accuracy as to both position and velocity. In general, mills and lathes are in this category.

Another way of categorizing NC machines is by the type of controller unit installed on them. The data processing unit on a conventional NC machine is a tape reader, which reads punched tape in 8-bit format; this is the oldest and is rarely used now. A CNC (computer numerical control) comes with a dedicated CPU, monitor, and local memory. Instructions can be generated directly from a computer and downloaded to the machine. Instructions can also be directly programmed on the CNC machine. A group of CNCs also can be networked and controlled by a central computer. This type of setup is called DNC (distributed numerical control), which is often seen on a shop floor.

Among all, the most common ways to classify NC machines is by the number of axes that the control drives simultaneously to move and rotate the cutter with respect to workpiece or vice versa. Two-axes NC refers to machines that control cutter motion simultaneously along two orthogonal directions in a plane; (i.e., *X*- and *Y*-axes). The cutter is independently controlled along the third axis, usually the *Z*-axis. *Z*-axis control is parallel to the normal direction of the *X*-*Y* plane. Machine supporting processes, such as turning, drilling, punching, and profiling, belong to a two-axis NC.

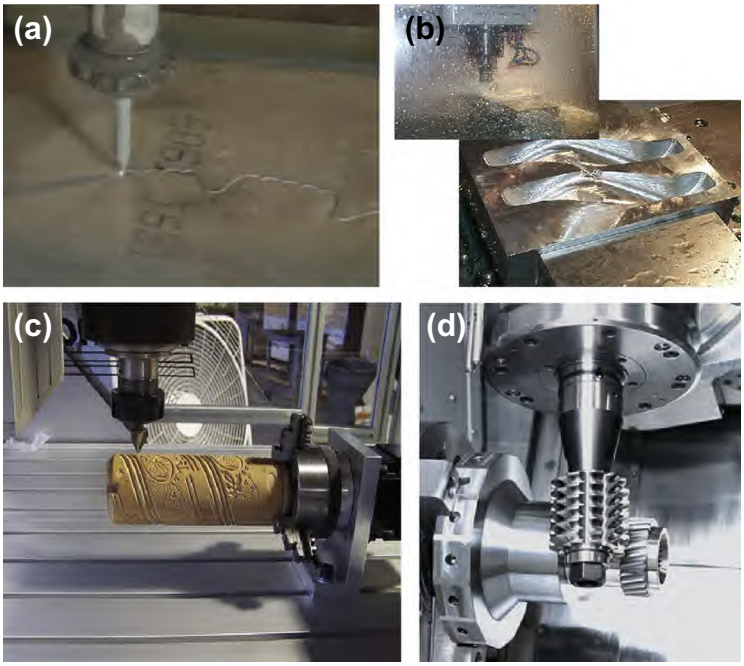


FIGURE 11.5

CNC machines of 2 to 5 *axes*: (a) water jet (Dix Metals, www.youtube.com/watch?v=crgujRcyhhE&feature=player_embedded), 2 *axes*, (b) HAAS 3-axis CNC mill, (c) 4th axis rotary table from Techno, (d) gear cutting using special 5-axis mill (www.amtek-precision.com).

In addition, a plasma cutter, such as PlasmaCAM (see images at www.plasmacam.com/indexfla.php), and a waterjet (see youTube video: www.youtube.com/watch?v=crgujRcyhhE&feature=player_embedded) are popular 2-axis NC machines.

Three-axis motion is most common in many aspects. The cutter is generally controlled in three principal directions of the Cartesian coordinate system simultaneously. A three-axis machine is able to cut not only pockets, profiles, and holes but also sculpture (or freeform) surfaces, often seen in mold or die manufacturing. For example, an ejector die shown in Figure 11.5(b) was machined using a HAAS 3-axis mill.

An add-on 4th axis rotary table can convert an existing 3-axis mill into a full 4-axis CNC machine. The rotary table allows spindle access to the workpiece from various angles in one setup that might take several setups with a conventional 3-axis machine (as shown in Figure 11.5(c)).

A 5-axis CNC mill in general provides simultaneous motion control in three linear directions (*X*-, *Y*-, and *Z*-axes) and two rotations, which usually are C (about *Z*-axis) and A (about *X*-axis), or C and B (about *Y*-axis). Such powerful machines support manufacturing a diverse range of components, such as gear (Figure 11.5(d)) and turbine blades (see image at www.thesurfacegrinder.com), to varying degrees of complexity and tolerances for high-precision industries.

11.2.2 BASIC CONCEPT OF PART PROGRAMMING

NC part programming consists of planning and documenting the sequence of processing steps to be performed on an NC machine. The outcome is an NC program used to machine a desired part. An NC program describes the sequence of actions of the controlled NC machine, which include, but are not limited to, the following:

- Tool movements, including direction, velocity, and position
- Tool selection, tool change, tool offsets, and tool compensation
- Spindle rotation direction and spindle rotation speed
- Cutting speed for different sequences
- Application of cutting fluids.

All actions must be specified in the NC program as a set of blocks (typically, hundreds or thousands). A block is a group of words terminated by the end-of-block character (usually a carriage return, CR). Figure 11.6 shows a sample block, which performs a cut from current cutter location to $X = 2.35$ and $Y = -0.475$ along a straight line at a feedrate of 5 in./min. In this sample block, there are seven NC words; each word consists of an alphabet address character (also called identifier) followed by a numeric character. In the sample block, N100 is simply a sequence number of the block. G01 and G41 are NC words that prepare cutter motion, where G01 specifies a linear motion and G41 turns on cutter radius compensations from the left. *X*2.35 and *Y*−0.475 specify the next cutter location. F5.0 specifies the feedrate, and M07 is a miscellaneous function, which turns a flood coolant on. Lists of sample identifiers and NC words are given in Appendices 11A and 11B, respectively, at the end of this chapter.

In general, there are three methods for creating NC programs: manual, computer-assisted, and CAD/CAM (Figure 11.7). All methods aim at creating NC codes, also called machine control data or machine code data (MCD). Most MCD (or NC code) is common among NC machines. Some codes may be machine-dependent; that is, the same NC word could mean something very different from one CNC machine to another, which presents a potential pitfall in NC programming. The computer-assisted and CAD/CAM approaches (to be discussed in the next subsections) offer assistance from computer and software tools in generating NC codes that alleviates the burden of coding by hand to some extent. Both approaches generate cutter location data (CL data) to be translated to MCD through a post-processor.

Manual NC programming is the most straightforward. The MCD can be directly programmed and entered into the CNC machine for simple cuts; for example, see profile milling in Figure 11.8. In this

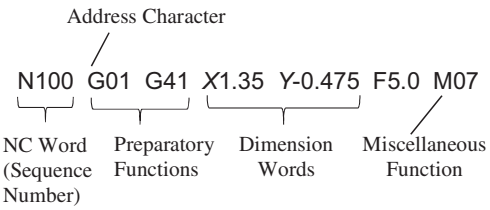


FIGURE 11.6

Sample NC block.

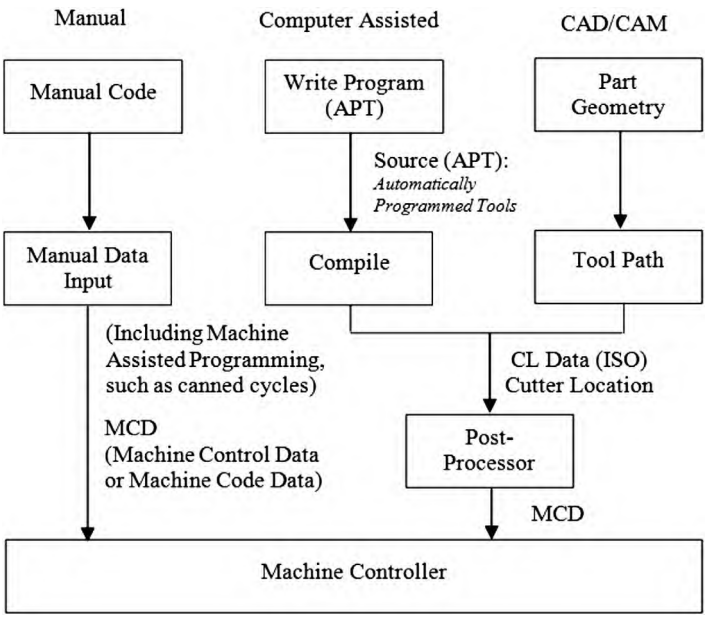
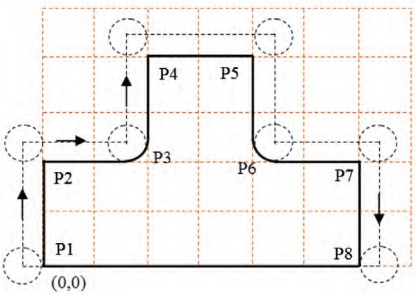


FIGURE 11.7

Approaches of NC programming.

(a)



(b)

N5 G90 G70 G40	(Absolute, inches, tool radius compensation off)
N10 M06 T13	(Tool change to Tool #13, 0.75" flat-end cutter)
N15 M03 S2000	(Spindle on clockwise at 2000 rpm)
N20 G00 X-0.375 Y0 Z1 M08	(Rapid to point #1: *X*–0.375, *Y*0, *Z*1, coolant on)
N25 G01 Z-0.5 F2.5	(Linear interpolation, Feed plunge to *Z*–0.5 at 2.5 ipm)
N30 Y2.375	(Feed move to point #2: *X*–0.375, *Y*2.375)
N35 X1.625	(Feed move to point #3: *X*1.625, *Y*2.375)
N40 Y4.375	(Feed move to point #4: *X*1.625, *Y*4.375)
N45 X4.375	(Feed move to point #5: *X*4.375, *Y*4.375)
N50 Y2.375	(Feed move to point #6: *X*4.375, *Y*2.375)
N55 X6.375	(Feed move to point #7: *X*6.375, *Y*2.375)
N60 Y0	(Feed move to point #8: *X*6.375, 0)
N65 G00 Z1	(Rapid retract)
N70 X0 Y0 M09	(Rapid to *X*0 *Y*0, coolant off)
N75 M05	(Spindle off)
N80 M30	(End of program)

FIGURE 11.8

A profile milling example: (a) desired toolpath and (b) NC codes with explanations.

sample of profile milling, a cutter of 0.75 in. diameter is cutting an 8 in. × 5 in. × 1 in. aluminum workpiece along a prescribed profile; the following is a desired toolpath:

- Tool start position: 0,0,1
- Move the cutter to –0.375,0,1
- Spindle speed 2000 rpm, feedrate 2.5 ipm

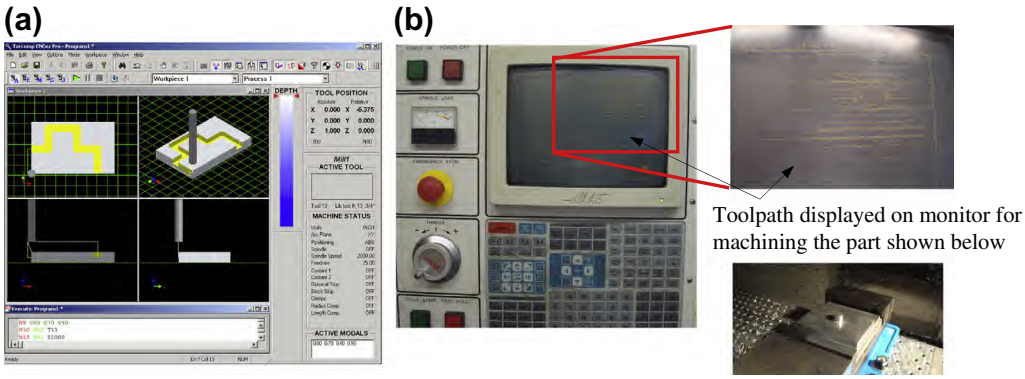


FIGURE 11.9

NC code verification using: (a) CNCezPro (www.cncezpro.com) and (b) HAAS 3-axis mill (www.haas.com).

- Plunge into workpiece 0.5 in. deep
- Cut along outside of profile formed by P1 up to P8
- Retract to 6.375,0,1 (next to P8)
- Move the cutter back to the start position.

Although simple and straightforward, manual programming at the MCD level is not necessarily the best. First, it is often time consuming to figure out cutter locations based on part geometry. Usually, a few predefined programs (canned cycles, such as hole-drilling) come with the CNC unit, which makes manual part programming a little easier. Second, the MCD is limited in supporting complicated machining work (e.g., contour surface milling), in which cutter locations on the surface must be accurately identified. Third, although MCD commands and formats are fairly well standardized, differences do exist between machine manufacturers (or the controller employed). As a result, an NC part program that works on one machine may not work on another.

It is critical that the MCD be verified before conducting actual cutting. This can be done by either using a third-party NC software tool, such as CNCezPro (Figure 11.9 (a)), or on the monitor screen of the CNC machine (e.g., HAAS 3-axis mill shown in Figure 11.9(b)).

11.2.3 COMPUTER-ASSISTED PART PROGRAMMING

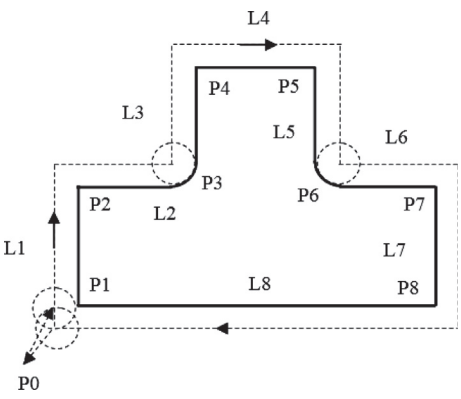
Computer-assisted part programming methods are faster and more reliable than manual programming techniques. There are a variety of forms of them. The common feature of these programs is that the part and machining paths are not defined directly with G-code but through English-like statements or through interactive graphic instructions. It consists of a series of English-like geometry and motion statements, which are used to define the workpiece and machine toolpath. A widely used language is Automatically Programmed Tools (APT) and its variations. Its instructions can be compiled and converted into G-code programs. Readers may refer to Chang et al. (1998) for a detailed discussion of APT.

APT uses language statements to define part shape and tool motion as well as machine tool-dependent data (e.g., feedrates and spindle speeds). The general procedures of generating APT source codes involve the following steps:

1. Identify part geometry.
2. Identify cutter motions, feeds, speeds, and cutter parameters.
3. Code the geometry, cutter motions, and general machine instructions into the part programming languages. The code is known as source.
4. Compile or process the source to produce the machine-independent list of cutter movements and auxiliary machine control information, known as the cutter location data file (or CL data).
5. The CL data file contains (mainly) details of cutter moves, either as a series of absolute linear GOTO moves or relative GODLTA moves. Note that the CL data are different from APT. The CL data are defined by the International Organization for Standardization (ISO).
6. Postprocess the CL data to produce MCD for the particular target CNC machine.
7. Transmit the MCD to CNC machine and verify it.

A major advantage of APT is that it has developed into an accepted standard for machine tools, in addition to alleviating the burden of coding at the very basic level. A prime disadvantage of APT is that it uses English-like commands to define geometry instead of the much more convenient graphical methods. Note that the APT-type programming approach is being gradually replaced by the more advanced CAD/CAM approach.

(a)



(b)

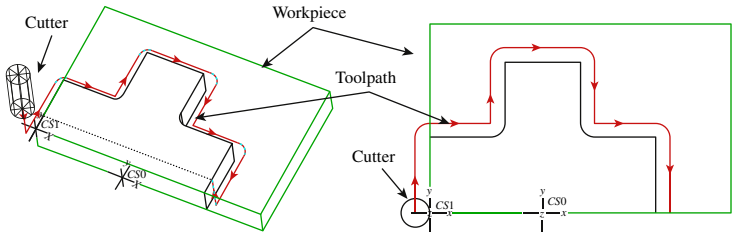
```
PARTNO/EXAMPLE                  PL1 = PLANE/0,0,–0.5,6.0,0,–0.5, 0,2.0,–0.5
MACHIN/MILL5                    SPINDL/2000
CUTTER/0.75                     FEDRAT/2.5
P0 = POINT/–1,–1,0              COOLNT/ON
P1 = POINT/0,0,0                FROM/P0
P2 = POINT/0.0,2.0,0            GO/TO, L1, TO, PL1, ON, L8
P3 = POINT/2.0,2.0,0            GOLFT/ L1, PAST, L2
P4 = POINT/2.0,4.0,0            GORGT/ L2, TO, L3
P5 = POINT/4.0,4.0,0            GOLFT/ L3, PAST, L4
P6 = POINT/4.0,2.0,0            GORGT/ L4, PAST, L5
P7 = POINT/6.0,2.0,0            GORGT/ L5, TO, L6
P8 = POINT/6.0,0,0              GOLFT/ L6, PAST, L7
L1 = LINE/P1, P2                GORGT/ L7, PAST, L8
L2 = LINE/P2, P3                GORGT/ L8, PAST, L1
L3 = LINE/P3, P4                RAPID
L4 = LINE/P4, P5                GOTO/ P0
L5 = LINE/P5, P6                COOLNT/OFF
L6 = LINE/P6, P7                END
L7 = LINE/P7, P8                FINI
L8 = LINE/P8, P1
```

FIGURE 11.10

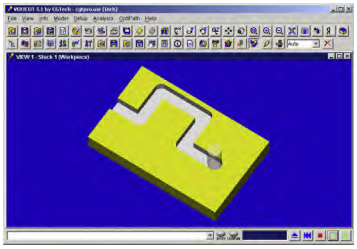
The profile milling example—APT approach: (a) desired toolpath and geometry entities identified and labeled and (b) APT source codes.

The same profile milling discussed earlier is employed as an example to illustrate APT programming. As shown in Figure 11.10(a), the geometry of the contour profile is identified and labeled as points (P0 to P8) and line segments (L1 to L8). These are specified in the APT codes shown in Figure 11.10(b). The motion statements, which describe the toolpath in relation to the part geometry, are given in the codes, such as GO/TO, GOLFT (toolpath to the left of the part), and GORGT, RAPID

(a)



(b)



(c)

```
PARTNO / PLATE
MACHIN / UNCX01, 1
UNITS / INCHES
LOADTL / 1
SPINDL / RPM, 2000.000000,  CLW
COOLNT / FLOOD
RAPID
GOTO / -0.3750000000, 0.0000000000, 1.0000000000
FEDRAT / 2.500000,  IPM
GOTO / -0.3750000000, 0.0000000000, -0.5000000000
GOTO / -0.3750000000, 2.0000000000, -0.5000000000
CIRCLE / 0.0000000000, 2.0000000000, -0.5000000000,  $
0.0000000000, 0.0000000000, -1.0000000000,  0.3750000000
GOTO / 0.0000000000, 2.3750000000, -0.5000000000
GOTO / 1.6250000000, 2.3750000000, -0.5000000000
GOTO / 1.6250000000, 4.0000000000, -0.5000000000
CIRCLE / 2.0000000000, 4.0000000000, -0.5000000000,  $
0.0000000000, 0.0000000000, -1.0000000000,  0.3750000000
GOTO / 2.0000000000, 4.3750000000, -0.5000000000
GOTO / 4.0000000000, 4.3750000000, -0.5000000000
CIRCLE / 4.0000000000, 4.0000000000, -0.5000000000,  $
0.0000000000, 0.0000000000, -1.0000000000,  0.3750000000
GOTO / 4.3750000000, 4.0000000000, -0.5000000000
GOTO / 4.3750000000, 2.3750000000, -0.5000000000
GOTO / 6.0000000000, 2.3750000000, -0.5000000000
CIRCLE / 6.0000000000, 2.0000000000, -0.5000000000,  $
0.0000000000, 0.0000000000, -1.0000000000,  0.3750000000
GOTO / 6.3750000000, 2.0000000000, -0.5000000000
GOTO / 6.3750000000, 0.0000000000, -0.5000000000
GOTO / 6.3750000000, 0.0000000000, 1.0000000000
RAPID
GOTO / -0.3750000000, 0.0000000000, 1.0000000000
COOLNT / OFF
SPINDL / OFF
FINI
```

(d)

```
%
O0100
(1)
(01/09/03-19:39:23)
N0010T1M06
S2000M03
G00X-.375Y0.
G43Z1.H01M08
G01Z-.5F2.5
Y2.
G02X0.Y2.375I.375J0.
G01X1.625
Y4.
G02X2.Y4.375I.375J0.
G01X4.
G02X4.375Y4.I0.J-.375
G01Y2.375
X6.
G02X6.375Y2.I0.J-.375
G01Y0.
Z1.
G00X-.375
M30
%
```

FIGURE 11.11

The profile milling example—CAD/CAM approach, (a) solid models and toolpath, (b) milling simulation, (c) CL data file, and (d) MCD (or tap file).

(rapid retract). Auxiliary statements, such as COOLNT ON (turn on coolant), FEDRAT (feedrate), SPINDL (spindle speed), are also specified.

11.2.4 CAD/CAM APPROACH

The CAD/CAM approach is the most popular and advanced approach in generating CNC codes. The process starts with creating a reference (or design) model and workpiece, and then assembling them in assembly mode. For example, the design model and workpiece of the profile milling discussed before are assembled, as shown in Figure 11.11(a).

After the solid models are assembled, users define manufacturing set up, including choosing a workcell (e.g., a 3-axis mill) and coordinate system (or machine zero). Then, users create a machining sequence (e.g., a profile-milling sequence). In the meantime, users choose a cutter and specify machining parameters (e.g., feedrate, spindle speed). At this point, a machining sequence is completely defined. CAD/CAM will generate a toolpath like that shown in Figure 11.11(a); the toolpath was created using Pro/MFG for this example.

The profile-milling sequence can also be simulated (e.g., in Vericut shown in Figure 11.11(b)). Note that Vericut (www.cgtech.com) is a third-party software that is integrated to Pro/MFG for CNC simulation and G-code verification. In addition, the toolpath generated can be output as a CL data file, as shown in Figure 11.11(c). The CL data can be converted to MCD using a proper post-processor that supports CNC machines at the shop floor. As an example, the MCD shown in Figure 11.11(d) is postprocessed for a HAAS CNC mill, which is also called a tap file.

In the rest of the chapter, we will focus on the CAD/CAM approach from a broader aspect—that is, virtual machining simulations in support of product design.

11.3 VIRTUAL MACHINING SIMULATIONS

A typical process of conducting a virtual machining simulation using CAD/CAM tools involves creating a design model (perfectly finished part), creating workpiece (a raw stock for machining), choosing manufacturing set up, defining machining sequences, generating toolpath, checking the results, and post-processing the toolpath for CNC codes, as shown in Figure 11.12(a).

The manufacturing set up involves choosing a machine (e.g., a 3-axis mill) and defining a machining coordinate system (also called machine zero), usually on the front left corner of the top surface of the workpiece, for the NC sequence. The definition of a NC sequence includes selecting the tool and retract plane (the plane to which the tool retracts after a cut) and defining machining parameters (e.g., feedrate, spindle speed). After a complete sequence is defined, the toolpath can be generated in the form of CL data. Machining toolpath (Figure 11.12(c)), machining simulation (Figure 11.12(b)), and important machining sequence information, such as machining time, are generated by the CAD/CAM software. In addition, a post-processor can be chosen to generate MCD (machine control data)—M-codes and G-codes for a particular type of CNC machine—from the CL data.

In the following, we will use examples to illustrate the process of conducting virtual machining, including milling and turning operations. We will focus more on milling and just mention turning briefly. This is because NC programming for turning sequences is often carried out manually since the *X*- and *Z*-coordinates of the cutter locations can be more easily acquired. Creating NC codes through

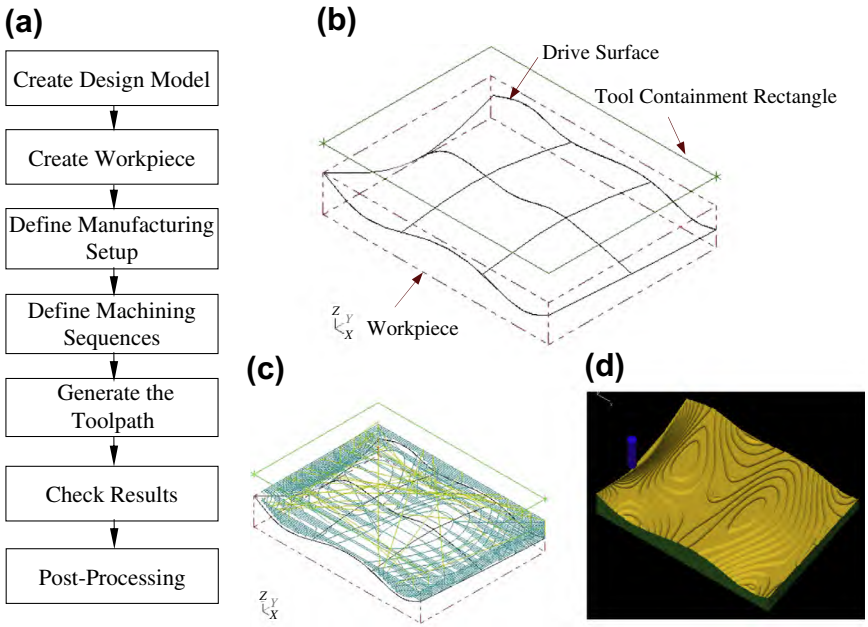


FIGURE 11.12

Virtual machining: (a) a typical process, (b) design model with workpiece, (c) machining toolpath, and (d) machining simulation.

virtual machining is sometimes overkill. However, from a design perspective, it is essential that designers incorporate all NC operations when addressing manufacturability of the product, including detecting potential manufacturing-related issues and coming up with an overall machining time estimate for design trade-off if necessary.

11.3.1 BASIC MACHINING SIMULATIONS

Basic machining operations refer to milling sequences involving simple geometric features such as profile milling, hole-making, pocket milling, and so on. These basic sequences can be performed using a standard 3-axis mill.

For a full-scale CAD/CAM system, such as Pro/MFG or CAMWorks, users will have to create 3D solid parts for the design model (and usually workpiece as well) in order to proceed. For software tools of CAM emphasis, such as Mastercam, a simple 2D wireframe is often sufficient to support basic machining operations. In this section, we will use a simple example shown in Figure 11.13(a) to discuss numerous aspects of the basic machining operations. This example involves profile milling, pocket milling, and hole-making. Figure 11.13(b) shows the toolpaths of the three NC sequences.

Using a CAM software, such as Mastercam, a 2D wireframe that describes the geometry of the outer profile, pocket, and holes (Figure 11.13(a)) is sufficient for carrying out a virtual machining operation. The size of the design model is 6 in. × 4 in. Both corner radius and diameter of the four holes are ¾ in. The radii of the large and small semicircles of the center pocket are 1 in. and ½ in.,

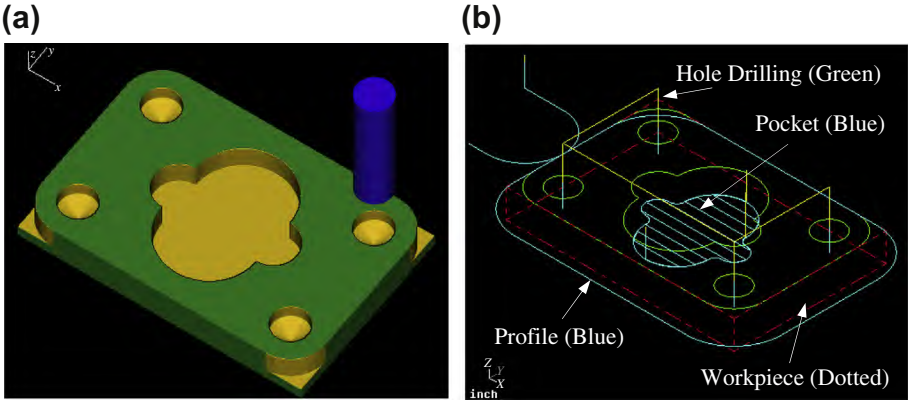


FIGURE 11.13

Basic machining sequences: (a) machining simulation and (b) machining toolpaths.

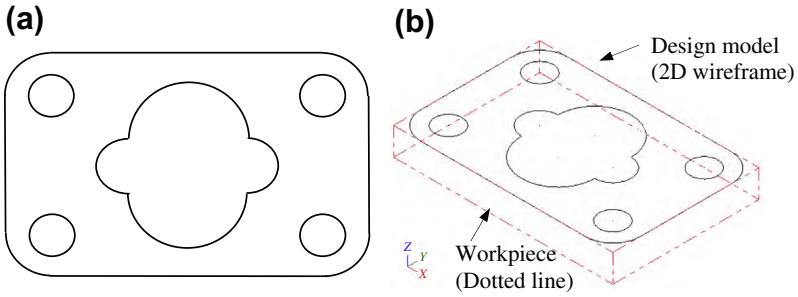


FIGURE 11.14

Basic machining example: (a) 2D wireframe design model, and (b) with workpiece in isoview.

respectively. A rectangular block of 6 in. × 4 in. × 0.75 in. is added to the wireframe as a workpiece, as shown in Figure 11.13(b). Note that the straight lines and the circular arcs in Figure 11.14(a) are created by entering end point locations, center point locations, radius, and so on, instead of creating solid features as those of CAD tools. While entering the point locations, a coordinate system was referred to. This coordinate system is implicitly defined as the reference, so called the machine zero, to which all CL data of the toolpath refer.

An end mill of 1.0 in. diameter is chosen for the first NC sequence, which is a profile milling (called contour milling in Mastercam). Some software (e.g., Mastercam) provides a detailed view on the geometry of the cutter, such as the one shown in Figure 11.15(a). Other software (e.g., Pro/MFG) requires users to enter basic parameters—for instance, cutter diameter and length—to define a cutter. In general, a cutter is represented as a cylinder in machining simulation for visual; for example, see the cutter of Mastercam shown in Figure 11.15(b). Note that Mastercam provides a predefined cutter library (Figure 11.15(c)) which helps users pick suitable cutters for machining. In practice, one will

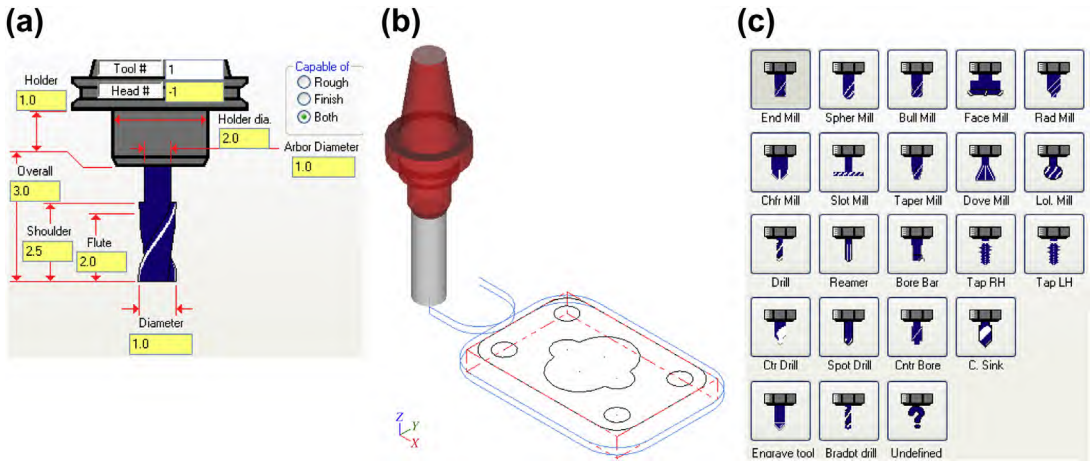


FIGURE 11.15

Selecting a cutter: (a) detailed geometry of an end mill in Mastercam, (b) cutter in virtual machining simulation, and (c) prescribed tool library in Mastercam.

have to check the mechanist at the shop floor to ensure that the cutters selected for virtual machining correspond to those on the tool turret in the CNC machine.

Several important cutting parameters are required for conducting profile milling. In this example, the (overall) *depth of cut* required for the profile milling is set to ½ in. Recall that the thickness of the workpiece is ¾ in. Therefore, there will be a layer of ¼ in. material intentionally left uncut after completion of the profile pass. The maximum *step depth*, which defines how thick the cutter trims around the workpiece in one pass, is set to ¼ in. in this example. Note that step depth is highly related to the size of the cutter among other factors. In general, a cutter of larger diameter has more strength to plunge deeper into the workpiece. A cutter of stronger material (e.g, cutter of carbide inserts) allows for a larger step depth. The rule of thumb is that the step depth has to be less than the radius of the cutter. Since the (overall) depth of cut is ½ in. and the maximum step depth is ¼ in., it will take two passes for the cutter to go around the workpiece to complete the profile milling.

Note that the extra steps—lead in and lead out, respectively—when the cutter approaches and leaves the workpiece are often added to the profile toolpath, as shown in Figure 11.16(a). Once the toolpath is generated, machining simulation can be shown like that of Figure 11.16(b) in, for example, Mastercam.

One practical issue that must be taken into consideration for profile milling is mounting the workpiece to the workbench (or jig table) of a CNC machine. Notice that in some cases, the cutter may not be able to cut a complete path around the workpiece without colliding with the clamps that hold the workpiece. Clamping and fixtures are an important issue in practice, which will be discussed in Section 11.4.

The second NC sequence is pocket milling. For a pocket milling, the cutter first plunges into the workpiece, then moves around to make a cut. The cutter usually moves in one direction, steps sideways, and cuts in the reverse direction. The cutting pattern is similar to mowing the lawn in your backyard, where we step sideways at the end of each pass before reversing the mowing direction. This

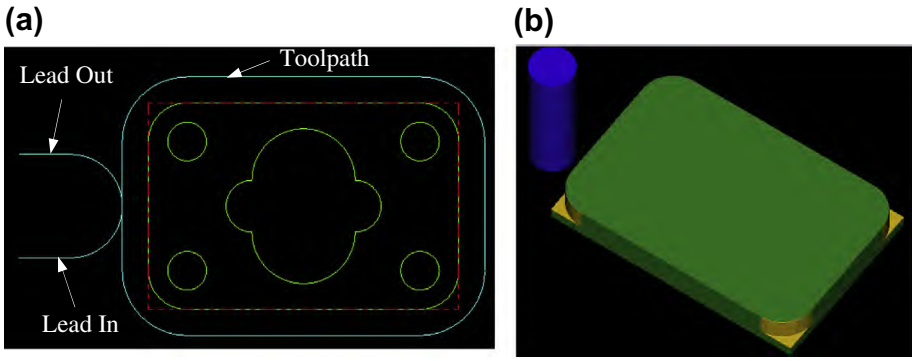


FIGURE 11.16

Profile milling sequence: (a) toolpath and (b) milling simulation.

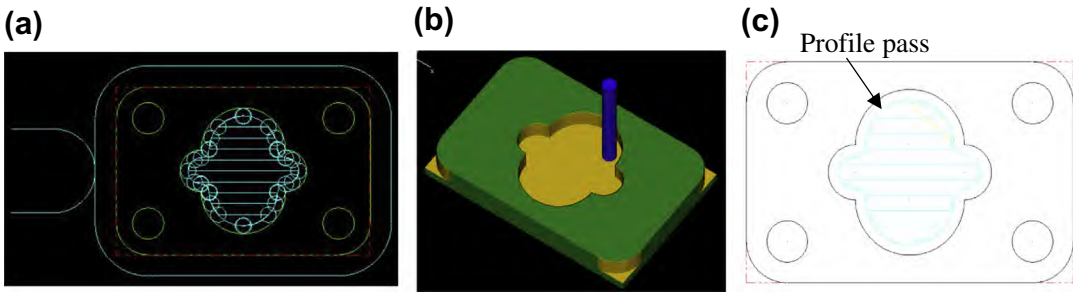


FIGURE 11.17

Pocket milling sequence: (a) toolpath, (b) milling simulation, and (c) profile cut.

is one of the available *scan patterns* in CAM, called zigzag. Other scan patterns include spiral, among others, where the cutter starts roughly from the center of the pocket and follows a spiral-like path and gradually moves outward.

The amount of step sideways is called *stepover*. It is obvious that stepover has to be less than the cutter diameter in order to make a clean cut. As a rule of thumb, the stepover is usually less than 80% of the cutter diameter in practice. After performing the scan passes, another profile pass is added to move the cutter around the wall of the inner pocket boundary to clean up the remaining material along the boundary, as shown in Figure 11.17(a). In this example, the cutter chosen is a ⅜ in. end mill in which the step depth chosen is $^1/_6$ in. If we are cutting a ½ in. deep pocket, it will take eight sets of complete scan and profile passes along the inner wall, each $^1/_6$ in. deep, to complete the pocket milling.

Also, in order to ensure a smooth inner pocket boundary, it is good practice to leave a thin material layer around the inner boundary wall of the pocket, and add a profile cut at the end to ensure a smoother pocket boundary, similar to that of Figure 11.17(c).

The third NC sequence is hole-making. There are four blind holes of ¾ in. diameter at the corners. Creating virtual machining for hole-making is straightforward. It is obvious that the drill you pick must

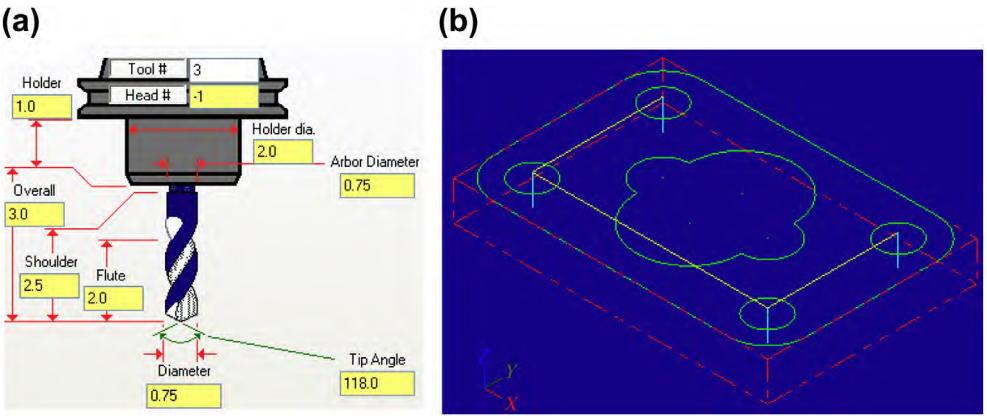


FIGURE 11.18

Hole-making NC sequence: (a) detailed geometry of a drill in Mastercam and (b) toolpath.

have the same diameter as those holes, or a drill of slightly smaller size with follow-on boring or reaming operations. In this case, a ¾ in. drill (Figure 11.18(a)) is chosen. The toolpath generated is shown in Figure 11.18(b).

All three NC sequences can be easily combined—toolpath and machining simulations can be like those shown in Figure 11.13(a) and (b), respectively. In addition, the toolpath can be converted into G-codes and M-codes, as discussed earlier.

11.3.2 ADVANCED MACHINING SIMULATIONS

Geometrically complex parts are commonly found in the automotive and aerospace industries where molds and dies are manufactured for production. The time spent to manufacture molds or dies of different sizes and complexities ranges from 1200 to 3800 hours (Sarma and Dutta, 1997). Considerable time (49–72%) is spent in contour surface milling for the design surfaces (the surfaces to be machined) of molds and die. Design changes in structural geometry due to functional considerations may affect manufacturing cost significantly, and vice versa. Among different types of machining operations, contour surface milling is the most critical for mold and die machining since surface milling directly affects the quality and accuracy of the design surface.

In practice, a machining operation for cutting a die or mold, or any parts with sculpture (or freeform) surfaces, involves four machining sequences. First, a face milling will clean up the mating surface for the die. The second sequence is usually a volume milling that removes material in the cavity at a fast speed. Often, a relatively large cutter is employed with a higher feedrate to remove material as fast as possible without regard to surface finish. Next is a local milling, which is essentially a smaller-scale volume milling that removes the remaining small chunk of material left from the previous volume milling sequence. In general, a smaller cutter is employed for local milling. The last NC sequence is contour surface milling—a finish cut; this is considered the most important machining sequence for sculpture surfaces since the scallop height of the uncut material remaining on the machined surface determines the surface quality of the machined part. In addition, common practice

suggests a small layer of stock should remain on the surface of the cavity from volume milling and local milling in order to eliminate tool marks.

Existing commercial virtual machining tools, such as Pro/MFG, CAMWorks, and Mastercam, support contour surface milling using various methods (e.g., isoparametric, constant curvature, and constant scallop height). In this section, we will assume Pro/MFG for machining a part of a 16 in. × 10 in. × 4 in. block with a sculpture surface as shown in Figure 11.19(a). Note that the sculpture surface is lofted using four section sketches of respective circular arcs (Figure 11.19(b)). For this example, we will include a volume milling and a local milling, assuming a 3-axis mill. We will include a 5-axis contour surface milling for finish cut. The workpiece is a 16 in. × 10 in. × 4 in. block. The coordinate system *CS0* at the top left corner of the workpiece, as shown in Figure 11.19(c), is chosen as the machine zero for this model.

Two cutters are chosen for this example. A 1 in. end mill with corner radius ¼ in. (Figure 11.20(a)) is selected for volume milling (i.e., the rough cut). Note that such a cutter is often referred to as a *bull mill*. Several such cutters are commercially available; for example, four- and two-flute end mills are shown in Figure 11.20(a). Note that the sharp corner of an end mill is its weakest point. The corner radius design of a bull mill strengthens the end mill by reducing chipping and providing longer tool

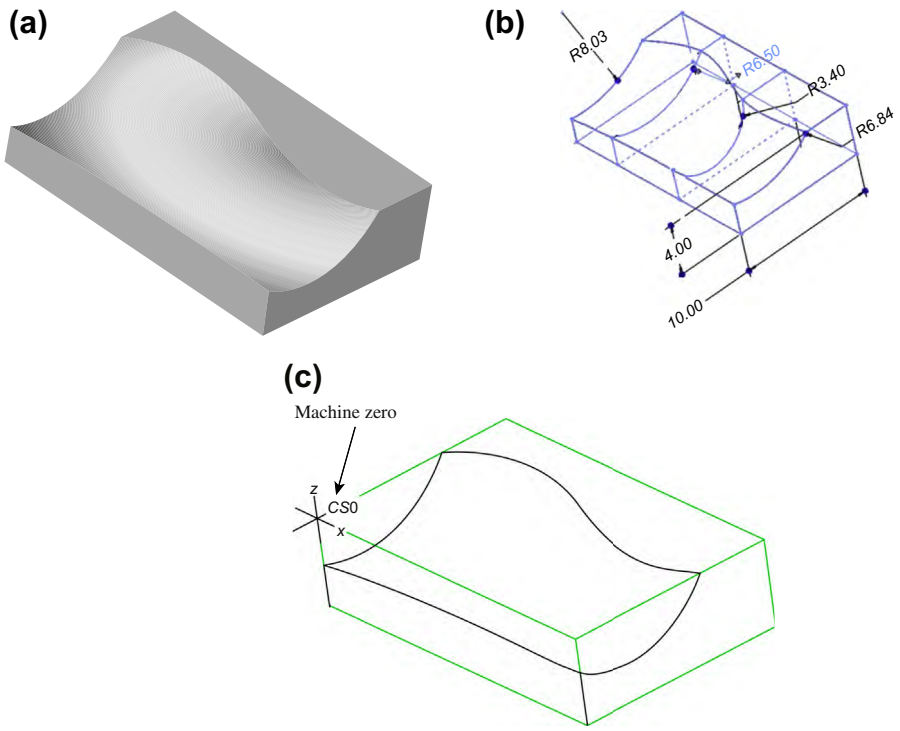


FIGURE 11.19

Sculpture surface: (a) solid model, (b) section sketches, and (c) with workpiece and machine zero.

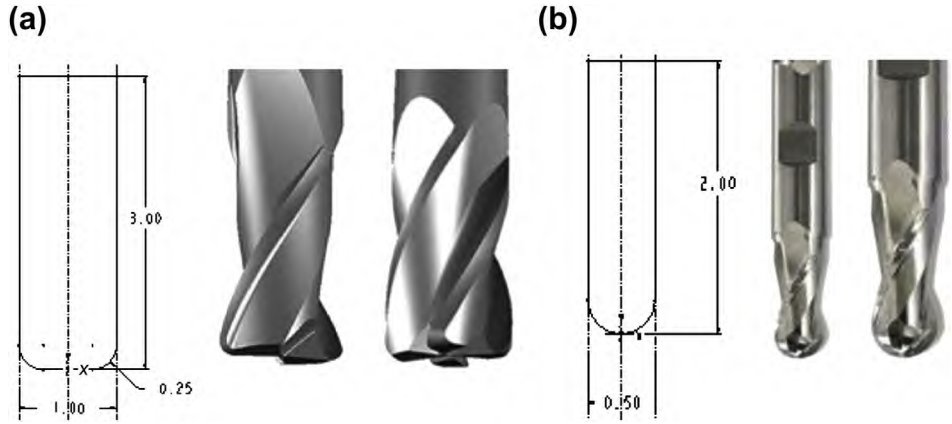


FIGURE 11.20

Cutters chosen for the machining operation, (a) 1 in. end mill with corner radius and (b) ½ in. ball cutter.

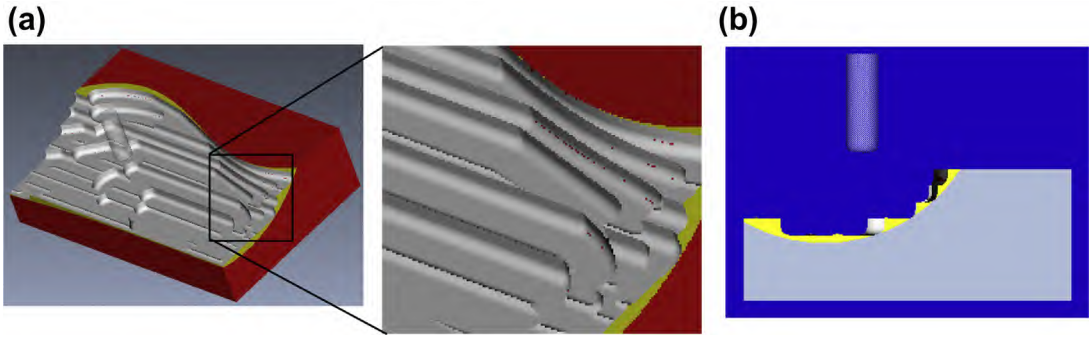


FIGURE 11.21

Volume milling NC sequence: (a) isoview and (b) end view.

life. In mold applications, both end and bull mill cutters have traditionally been used. In addition to the bull mill, a ½ in. ball nose cutter (Figure 11.20(b)) is employed for both local milling and contour surface milling. Note that a ball nose cutter tends to leave less tool marks on the machined surface, thus leading to a better surface finish.

For volume milling, the step depth and stepover are 0.5 in. and 0.9 in., respectively. The scan pattern is back and forth along the longitudinal direction (x-axis of *CS0*). The resulting machined surface is shown in Figure 11.21, in which material remains on the surface and requires a following NC sequence for further clean up. The machining time estimated by Pro/MFG is 37 minutes, assuming a feedrate of 20 in./min. Determining an adequate feedrate for a machining sequence is critical in practice. A fast feedrate will reduce overall machining time; however, it is subject to a possible rough surface finish and a higher probability of cutter breakage and shorter tool life. Feedrate will be discussed in more detail in Section 11.4.

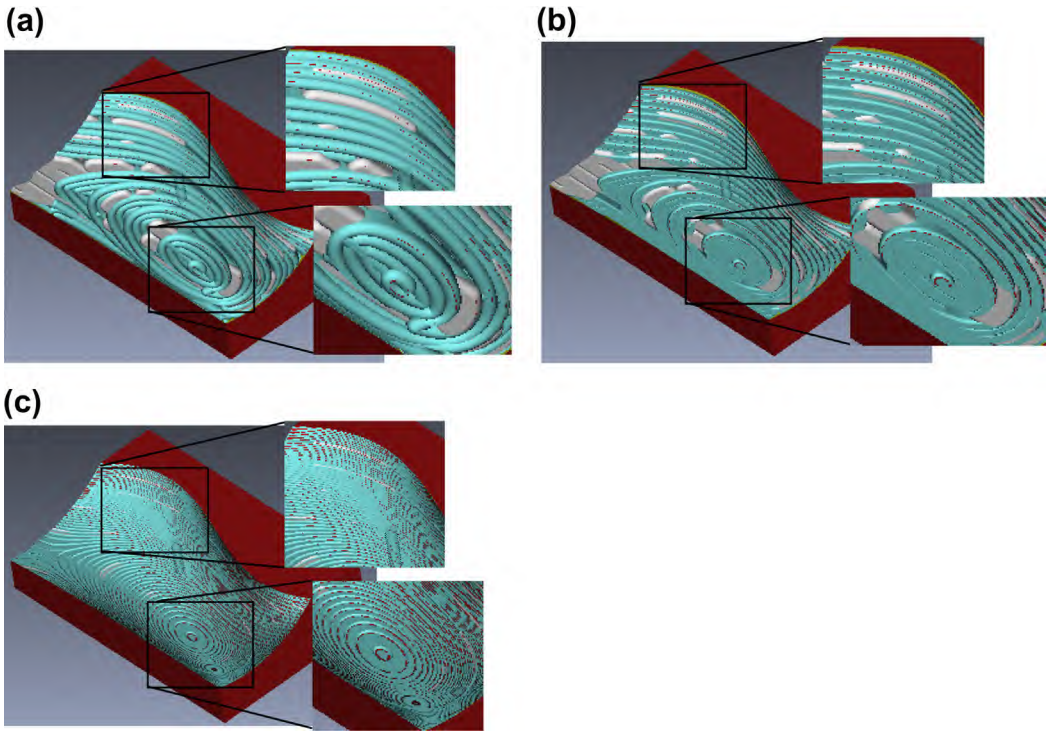


FIGURE 11.22

Machined surface after the local milling sequence: (a) ball nose cutter with stepover 0.4 in. and step depth 0.2 in., (b) end mill of corner radius 0.1 in. with stepover 0.4 in. and step depth 0.2 in., and (c) end mill of corner radius 0.1 in. with stepover 0.4 in. and step depth 0.05 in.

For local milling, the step depth and stepover are 0.2 in. and 0.4 in., respectively. Both are smaller than those of volume milling. This is mainly because a smaller cutter is employed. Also, a smaller feedrate, 5 in./min., is chosen for this sequence. The scan pattern is spiral, which is determined by Pro/MFG. The resulting machined surface is shown in Figure 11.22(a), in which the surface is smoother but still requires further clean up and polish. The machining time for the local milling is estimated by Pro/MFG as 137 minutes partly due to a smaller feedrate.

The machined surface is certainly not acceptable for practical applications. Before carrying out a contour surface milling, is there any way to improve the quality of the machined surface? Is the current setup for local milling the best? There are several possibilities that may provide a better result. First, an end mill of the same diameter (½ in.) but a smaller corner radius 0.1 in. (vs. ¼ in.) requires about the same machining time. Comparing with the first case shown in Figure 11.22(a), this bull mill provides a machined surface with a slightly better surface—especially in the area closer to the center of the spiral pattern (Figure 11.22(b)). This is mainly due to the smaller surface curvature closer to the center of the spiral pattern, in which a bull mill leaves less material than that of a ball nose cutter.

How about step depth and stepover? Which parameter will impact more on the surface quality? In general, it depends on the geometric shape of the design surface. For a surface of high curvature,

a smaller step depth reduces the staircase type surface. For a surface of small curvature, one can afford to choose a larger stepover to reduce the machining time and yet lead to a relatively good surface finish. Smaller stepover and smaller step depth will certainly give a better surface finish; however, it may require substantially more machining time to complete the cut. For example, Figure 11.22(c) shows a much smoother machined surface created by using a smaller step depth, 0.05 in., in which the machining time jumps to more than 600 minutes, assuming the same feedrate. Note that in practice, one may increase the feedrate when a smaller stepover and step depth are employed, resulting in a reduced machining time.

Although surface quality is much improved, spending 600 minutes for a local milling is probably too much. Plus, we will employ a contour surface milling to perform the finish cut, which serves the purpose of polishing the surface. For this example, we will stay with the first case (surface finish shown in Figure 11.22(a)) after completing the local milling sequence.

The third NC sequence is contour surface milling for a finish cut using a 5-axis mill. A step size of 0.4 in. and the same feedrate, 5 in./min., are chosen for this sequence. As mentioned earlier, there are several options in generating a toolpath for the contour surface milling. We choose the surface isoline option for toolpath generation, where one parametric coordinate of the surface is fixed at prescribed values in order to generate cutter contact curves that stay right on the surface (Figure 11.23(a)). Note that the space between neighboring tool passes is in general less than the stepover specified. More on

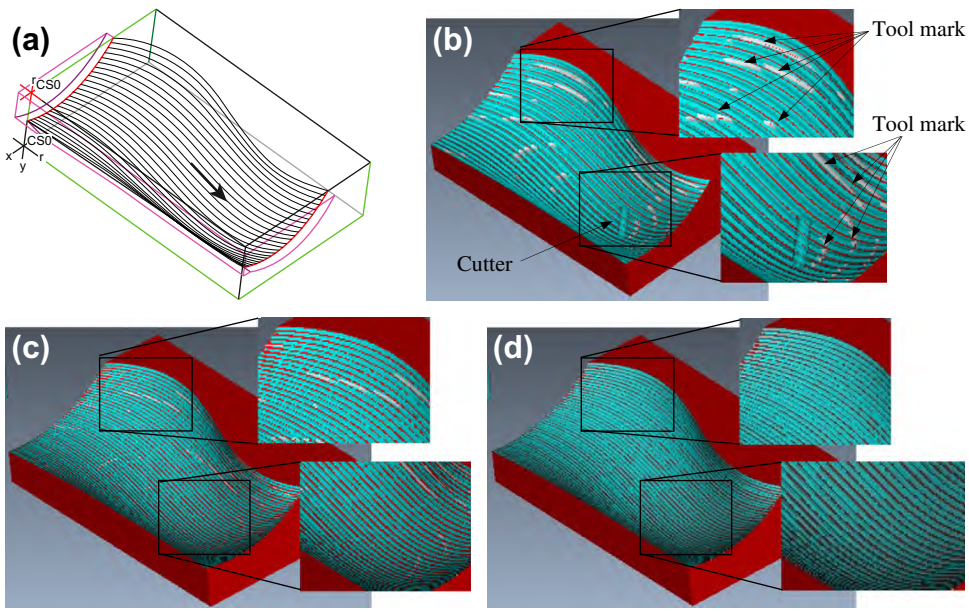


FIGURE 11.23

Contour surface milling: (a) toolpath for stepover 0.4 in.; (b) machined surface with stepover 0.4 in., tool mark visible; (c) machined surface with stepover 0.2 in, tool mark still visible; and (d) machined surface with stepover 0.4 in. and 0.05 in. stock allowed, tool mark removed.

the isoline option and toolpath generation are discussed in Chapter 12. The machining time for this contour surface milling is estimated as 85 minutes, based on a feedrate of 5 in./min. Note that in practice, a higher feedrate is often employed for the final finish cut since the material remaining on the surface is usually minimal.

The resulting surface by combining all three NC sequences can be seen in Figure 11.23(b). There are clear tool marks on the machined surface. These marks were left from the previous sequences, mainly from the rough volume milling. There are two ways to remove or minimize the tool marks. First, one may reduce the stepover for the contour surface milling. A smaller stepover will lead to a surface with smaller scallop height; therefore, a better surface quality with less visible tool marks (Figure 11.23(c)), in which the stepover is reduced to 0.2 in. However, the machining time increases about twice (increased from 85 to 160 minutes). We discuss more on the calculations of the scallop height in Chapter 12.

Another approach, and a better one, is to specify a small layer of stock material uncut (stock allowed) while performing volume and local milling sequences. The layer thickness should be slightly larger than the desirable scallop height. If such a stock material is specified in previous sequences, tool marks should be completely removed in theory. With a stock material of 0.05 in. and a smaller stepover (0.2 in.), the machined surface does not reveal any tool marks, as can be seen in Figure 11.23(d).

If combining all three NC sequences, the overall machining time is about 334 minutes (37 + 137 + 160), which is considered very good for machining such a sculpture surface.

11.3.3 TURNING SIMULATIONS

Part programming for turning is much simpler than that of milling. This is mainly because that NC code for turning only requires *X*- and *Z*-locations of the cutter. A cutter for a turning operation only moves on the *X*-*Z* plane, often following straight paths. For example, an area turning sequence that removes the material on the outside cylindrical and front-end surfaces of a cylindrical block is shown in Figure 11.24. Note that when translating the toolpath to NC codes, special attention may be required. If you are converting a toolpath created in Pro/MFG to HAAS CNC lathe, there are several modifications and/or extra steps that need to be carried out. For example, the CL data output for the G02 and G03 commands (circular interpolation) must use "R" values for radius. If "I" or "K" values are present, a setting for the post-processor will need to be altered in order to force the output with "R" values. Also, while selecting UNCL01.P11 to postprocess the CL data for HAAS lathe, this post-processor does not output the *X*, *Z*, *R* (*R* is the radius of circular interpolation in G02 and G03). In addition, the post-processor does not output the *F* (feedrate) values with a decimal point, which is required by the HAAS controller. Again, a setting for the post-processor will need to be altered to output CNC codes with the correct format.

Going over all the modeling and simulation work for a simple turning cut, such as the area turning shown in Figure 11.24(b), may not be the best option in terms of time and effort. Plus, there are additional setups in the NC post-process or that are usually required for valid NC codes. Therefore, manual coding is often more effective for the CNC lathe. Even when machining a sculpture surface on a turning operation, a toolpath can be defined along a planar curve, which is much easier to obtain manually than that of milling often dealing with a surface. As a result, part programming for turning operations is usually carried out by collecting cutter location data on the *X*-*Z* plane and manually composing NC programs.

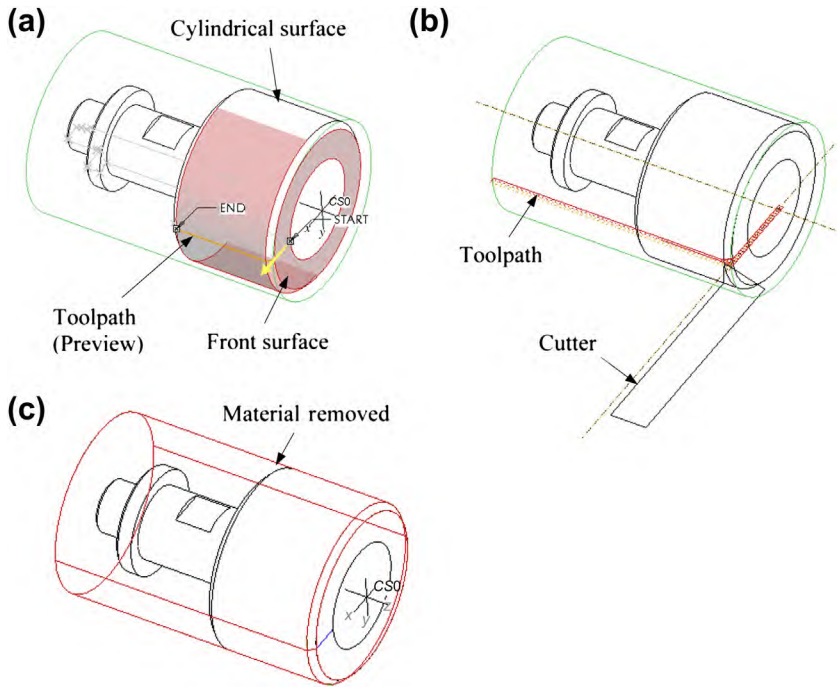


FIGURE 11.24

Area turning: (a) material to be removed on the cylindrical and front end surfaces, (b) turning toolpath, and (c) machined surface.

11.4 PRACTICAL ASPECTS IN CNC MACHINING

So far, our discussion has been focused on virtual machining—everything is carried out in a virtual environment. In a virtual environment, almost anything can be set up and simulated. Machining sequences that look good on a computer do not guarantee they can be implemented physically without any flaws. For example, a deep and narrow pocket may not be possible to cut since a small-size cutter is usually short and will not be able to reach deep enough to the bottom of the pocket. Also, for profile milling in a virtual environment, a cutter moves around the outer boundary of the workpiece without any problem. In reality, the workpiece must be clamped to the workbench (or jig table) of the CNC machine and the cutter must not collide with the clamps. What kind of clamp or fixture is suitable for mounting the workpiece for carrying out a profile milling sequence?

When a fixture is employed to mount the workpiece, will the cutter run into the fixture? Also, how does one determine appropriate feedrate and spindle speed? Which kind of cutters are suitable for a given workpiece material using selected NC sequences? Most such issues have been well addressed in NC workbooks or machinist handbooks (e.g., Oberg et al., 2012). We will discuss some of the common practical issues you may encounter in NC machining. The goal of this section is to provide you with some practical aspects in transition of virtual machining to practical NC operations.

There are several principles you may want to consider while designing a part to be machined. These principles usually lead to a part design that can be machined with fewer problems. First, designers should consider designing parts that can be made out of standard-size material stock. This will minimize the need to go through additional machining sequences that cut a standard stock to the desired size. Second, avoid deep pockets in parts; you may not find a cutter long enough to reach the bottom of the pocket. Even if you can, the long cutter may deflect or break during deep pocket cutting. Third, keep all fillet radii identical whenever possible and keep them larger than the radius of available cutters. A large cutter will not be able to cut a small fillet. Fourth, keep all hole sizes identical if possible and avoid odd-size holes. As a result, one or two drills are able to make all holes, which minimizes the frequency for tool change. Hole sizes should in general match those of available drills. Finally, avoid having to cut the part from multiple sides if possible. Certainly, compromises will always have to be made when designing a new part with machining considerations. However, staying with the preceding principles as much as possible will make the machining sequences more straightforward and usually less costly.

In the following, we will discuss three main issues: (1) choice of jigs and fixtures; (2) determination of machining parameters, including feedrate and spindle speed, as well as cutter selection; and (3) setting up NC machining.

11.4.1 JIGS AND FIXTURES

Mounting the stock material to the workbench (also called *feed table* or jig table) is often the most critical and time-consuming element of part machining. The stock material must be mounted in a way that all the tools are able to complete their paths without interfering with a fixture or the stock coming loose or moving.

The easiest way to mount stock is in a vise (Figure 11.25(a)). Once the vise is aligned and mounted to the workbench, the coordinate system for the part can easily be identified (more about this in Section 11.4.3). For cutting circular parts or placing bolt patterns at the end of shafts, a self-centering 3-jaw chuck (Figure 11.25(b)) will save time. Flat plate or sheet material can be clamped using the standard block, and a clamp set can be used to fix generic parts to the table (Figure 11.25(c)).

When machining odd-shaped parts, special jigs may have to be made. It is important to be patient in fastening the stock to the workbench. The following are a few examples that are employed to mount the workpiece.

FIGURE 11.25

Common jigs and fixtures: (a) machinists vise, (b) self-centering 3-jaw chuck, and (c) block and clamp setup.

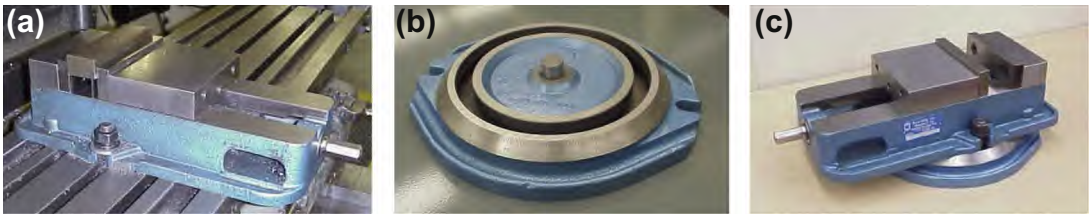


FIGURE 11.26

Milling machine vise: (a) plain vise, (b) swivel base, and (c) swivel base with vise.

The milling machine vise shown in Figure 11.26 is the most common type of workpiece-holding device used on the milling machine. The plain milling machine vise is used for holding a workpiece that has parallel sides. The vise is bolted directly to the table using the T-slots in the machine table (Figure 11.26(a)). The plain vise can be accompanied by a swivel base. The swivel base (Figure 11.26(b)) is graduated in degrees and allows the vise to swivel in the horizontal plane. The swivel base gives the vise a greater degree of versatility (Figure 11.26(c)), but should be avoided when doing heavy rough-cutting operations because it reduces the rigidity of the setup.

11.4.2 CUTTERS AND MACHINING PARAMETERS

The choice of cutters and workpiece material largely determine feedrate and spindle speed. Cutters can be grouped into two major categories: high-speed steel (HSS) and carbide inserts, as shown in Figure 11.27. HSS cutters maintain hardness and strength at elevated temperature and are less expensive. Two basic types of HSS cutters are commercially available: M-series (molybdenum type) and T series (tungsten type). M-series steels contain up to 10% molybdenum, with chromium, vanadium, tungsten, and cobalt. T-series steels contain 12 to 18% tungsten, with chromium, vanadium, and cobalt.

M-series steels have generally higher abrasion resistance than T-series, less distortion in heat treatment, and are less expensive. Carbide inserts endure high temperatures with high strength and

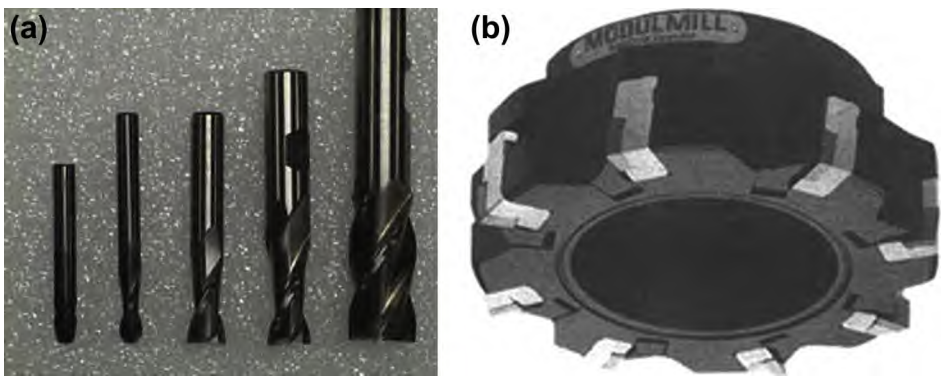


FIGURE 11.27

Cutters: (a) HSS cutters and (b) carbide insert.

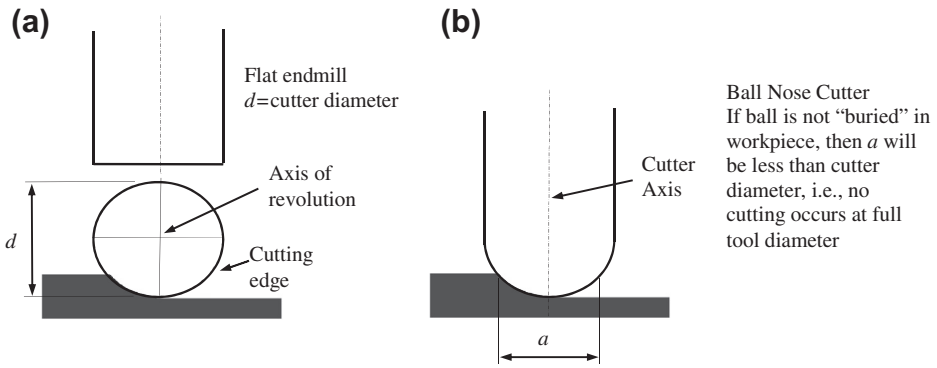


FIGURE 11.28

Cutters: (a) flat endmill and (b) ball nose cutter.

hardness, especially suitable for high-speed cutting. The tungsten carbide (WC) cuts nonferrous abrasive materials and cast irons. Titanium carbide (TiC) has higher wear resistance than WC but is not as tough. Titanium carbide cuts hard materials, mainly steels and cast irons.

From their geometric shape, there are flat endmill (Figure 11.28(a)), ball nose cutter (Figure 11.28(b)), and bull mill (Figure 11.20(a)). In general, endmills are suitable for face, profile, and pocket milling. A ball nose cutter is more suitable for sculptural surface milling since there is less tendency to gouge using a ball nose cutter.

Two of the most critical machining parameters are spindle speed and feedrate. Spindle speed N (rpm) is usually determined by the cutting speed V (in./min.); that is,

$$N = V/(\pi d) \tag{11.1}$$

where d is the cutter diameter (in.). In general, factors affecting the calculation of cutting speed include (1) the material being machined (steel, brass, tool steel, plastic, wood), (2) the material the cutter is made from (carbon steel, high-speed steel, carbide, ceramics), and (3) the economical life of the cutter; that is, the cost to regrind or purchase new, compared to the quantity of parts produced.

You may find recommended cutting speeds in machining textbooks or handbooks; for examples, see Tables 11.1(a) and (b). As shown in the tables, cutting speed is provided in a large range. This is because there are other factors to consider while determining cutting speed (e.g., type of the milling sequences). A rough cut usually requires a larger cutting speed, thus a higher spindle speed, in order to remove material fast. Different data sources may offer cutting speeds that differ by, sometimes, significant amounts. Numbers obtained from machining handbooks are a good starting point. You may have to make needed adjustments on-site.

Once an adequate spindle speed is determined, a feedrate f (in./min.) can be calculated using the following equation:

$$f = f_t N n \tag{11.2}$$

where n is the number of teeth on the cutter, and f_t is the feed per tooth (in.), as depicted in Figure 11.29. Factors that affect the feedrate include: (1) type of tool (e.g., a small drill vs. a large end mill), (2) surface finish desired, (3) power available at the spindle (to prevent stalling of the cutter or

Table 11.1 Recommended Cutting Speeds

(a) Recommended by Kalpakjian and Schmid (2010)

Workpiece Material	Cutting Speed (m/s)	
	HSS	Carbide Inserts
Aluminum alloys	1.5–6	10 +
Magnesium alloys	3–5	12 +
Copper alloys	0.3–1.5	1.5–7
Steels	0.1–0.7	0.5–4
Stainless steels	0.2–1	1–2
High-temp alloys	0.05–0.1	0.2–0.3
Titanium alloys	0.1–1	0.5–2
Cast irons	0.2–0.6	0.5–2

(b) Recommended by Krar et al. (2010)

Workpiece Material	Cutting Speed (m/min)	
	HSS	Carbide Inserts
Machine steel	21–30	45–75
Tool steel	18–20	40–60
Cast iron	15–25	40–60
Bronze	20–35	60–120
Aluminum	150–300	300–600

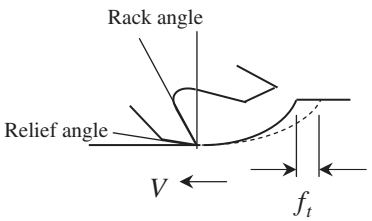


FIGURE 11.29

Feed per tooth.

workpiece), (4) rigidity of the machine and tooling setup (ability to withstand vibration or chatter), (5) strength of the workpiece (high feedrates will collapse thin-wall tubing), and (6) characteristics of the material being cut (e.g., chip flow depends on material type and feedrate).

The recommended feeds per tooth for high-speed steel and carbide inserts are listed in Tables 11.2(a) and (b), respectively. As shown, higher feed per tooth is recommended for carbide insert

Table 11.2 Recommended Feed per Tooth by Krar et al. (2010)

(a) High-Speed Steel

Workpiece Material	Face Mills		End Mills	
	in.	mm	in.	mm
Aluminum	0.002	0.55	0.011	0.28
Brass and bronze	0.014	0.35	0.007	0.18
Cast iron	0.013	0.33	0.007	0.18
Bronze	0.012	0.30	0.006	0.15
Tool steel	0.010	0.25	0.005	0.13
Stainless steel	0.006	0.15	0.003	0.08

(b) Carbide Inserts

Workpiece Material	Face Mills		End Mills	
	in.	mm	in.	mm
Aluminum	0.020	0.50	0.010	0.25
Brass and bronze	0.012	0.30	0.006	0.15
Cast iron	0.016	0.40	0.008	0.20
Bronze	0.012	0.40	0.008	0.20
Tool steel	0.014	0.35	0.007	0.18
Stainless steel	0.010	0.25	0.005	0.13

cutters while cutting hard materials such as steels. Also, higher feed per tooth is recommended for face milling than end milling. This is because more teeth are in contact with the workpiece in face milling; thus, a higher feedrate. While carrying out finish cut, the feedrate can be turned up if material remaining is minimal. Again, numbers obtained from machining handbooks only provide a good starting point.

11.4.3 SETTING A CNC SEQUENCE

In this subsection, we will go over a few important points in setting up an NC machining sequence at the shop floor. These are the things that one must be aware of when an NC program is generated and is ready to carry out the machining sequence using a CNC machine. Since there are wide varieties of machines, we will only discuss two key steps without referring to specific machines. The goal here is to provide you with some ideas when bringing the part program to a CNC mill, and to serve as a reminder that there are practical issues to go over before a successful machining sequence can be carried out on the shop floor.

After a workpiece is loaded and secured on the workbench, two key steps are followed: loading and setting up tools, and setting up work coordinate and machine zero.

FIGURE 11.30

Using edge finder to set up *X*-axis.

Loading cutters is one of the most important steps in setting up a CNC machining sequence. You may collect all the needed cutters and follow the machine instructions to load them to the mill. Once all tools are loaded into the tool changer the lengths of each must be measured. You will also need to measure cutter diameters, especially if tool radius compensations are included in your NC codes. Before measuring lengths, we first place a work coordinate (or machine zero) on the stock material in the same place that you have set it in CAM software (such as Pro/MFG or Mastercam). Machine zero is usually placed at the front left corner of the top surface of the stock. Usually, we load a drill chuck with an edge-finding tool (Figure 11.30) into the spindle, move the edge finder to the *X* position of the part using the jog handle, and zero the *X*-axis. We repeat the similar steps for setting up *Y*-axis.

The *Z*-axis will be set after all the tools needed for machining are loaded. Often the first cutter is advanced into the spindle. Then, we position the tool tip over the top of the *X* and *Y*-zero. We lower the tool and place a feeler gage (Figure 11.31) to set zero for *Z*-axis. Since the *Z*-axis is zeroed using the

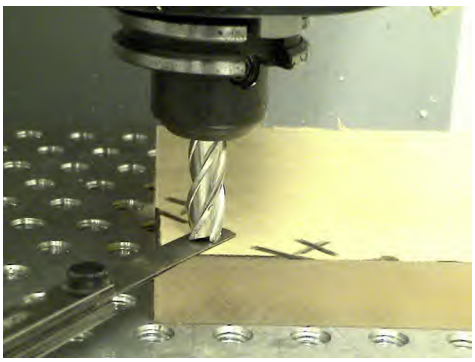


FIGURE 11.31

Z-axis offset and tool length offset.

first tool, the length offset for tool one should be zero. The thickness of the feeler gage must now be subtracted from the *Z*-axis height. The same steps will have to be repeated to measure or offset the tool lengths for all cutters loaded.

Once these two steps are completed, you are ready to load the program to the machine. Usually a CNC mill is networked with a PC. In that case, an NC program can be loaded to the mill very easily. Once a program is loaded it usually automatically becomes the selected program the mill will run. It is good practice to verify the toolpath on the monitor screen of the mill, like that shown in Figure 11.9(b), before running the program.

11.5 COMMERCIAL MACHINING SIMULATION SOFTWARE

A large number of commercial CAM software tools can currently be obtained. There is general-purpose CAM software that supports turning and milling sequences. There are also special-purpose codes that focus on narrow tasks (e.g., software for creating a toolpath only for engraving).

General-purpose software offers adequate geometric modeling capabilities. Most of them are tightly integrated with an existing CAD system. Roughly, the general-purpose CAM software can be categorized into three main groups in terms of how they relate to a CAD system. The first group includes CAM as one of the modules in a CAD system, such as Pro/MFG of Pro/ENGINEER, CAM module of CATIA CAD/CAM/CAE system, and CAM module of the NX Unigraphics. The second group refers to CAM as third-party software that is seamlessly integrated into commercial CAD software. This includes CAMWorks (www.camworks.com) integrated with SolidWorks, InventorCAM (www.inventorcam.com) with Autodesk Inventor, and so on. The third group includes stand-alone CAM software that is not associated with any CAD systems (e.g., Mastercam, SurfCAM (www.surfware.com), and GibbsCAM (www.gibbscam.com)).

Special-purpose codes, as mentioned earlier, focus on narrow tasks. Some of these codes include ArtCAM (www.artcam.com) that supports sign makers, woodworkers, engravers, and jewelers to design and manufacture 3D or 2D products from their artwork.

This section presents a brief overview of the commercially available CAM software. We will include a short description for one or two of the software from each group. The strengths, weaknesses, pros, and cons of these codes will be briefly discussed.

11.5.1 GENERAL-PURPOSE MACHINING SOFTWARE

Major CAD systems, such as Pro/ENGINEER, CATIA, and NX, are equipped with a CAM module that is part of the CAD and is seamlessly integrated with the respective CAD systems. As mentioned before, other major CAD systems (e.g., SolidWorks or AutoDesk) offer a CAM module that is developed by a third party and is seamlessly integrated with the CAD systems as well. All such CAM modules provide excellent virtual machining capabilities in support of a broad range of machining sequences, including profile milling, volume milling, surface milling, hole-making, text-engraving, and so on. These capabilities support users to create virtual machining simulation, generate a toolpath, and convert the toolpath into CNC codes, following the steps depicted in Figure 11.12(a). Development of software in this category essentially started with CAD and then extended to include CAM either through its own development and acquisition or via a third party.

The major advantage of these software tools is that they require a relatively short learning curve for new CAM users. This is mainly because the user is already familiar with the basic operations and procedures of using the CAD portion of the software. Learning CAM that is a built-in module of a CAD system the user is familiar with is relatively straightforward. However, using a CAD-centered CAM module, users must start with creating 3D solid models that represent design and workpiece, even for a simple cut such as hole drilling.

One thing worth noting is that CAMWorks offers a knowledge base and Automatic Feature Recognition (AFR) tools to help users in the machining processes. The AFR technology analyzes solid model geometry and identifies machineable features (e.g., holes, slots, pockets, and bosses). The knowledge base is a self-populating database that contains information about the cutting tools and the parameters used by the operator, as well as information regarding the cutting tools available on the shop floor. This database helps store the best practices at a centralized location in the tool room.

On the other hand, software, such as Mastercam, SurfCAM, and GibbsCAM (all general-purpose CAM software), were developed and marketed with the sole purpose of supporting virtual machining, which is essentially CAM-centered. These tools offer excellent capabilities for virtual machining, often even more general and versatile than those integrated in CAD systems. For example, Mastercam provides more options for generating a toolpath for contour surface milling, including the constant scallop height option that was not available in Pro/MFG.

In general, virtual machining can be carried out with these tools using wireframe, surface, or solid geometric entities. For example, a pocket milling or a profile milling only requires a simple wireframe sketch that describes the geometry of the pocket and profile boundary of the part. This is much simpler and requires less effort compared with that of using a CAD-centered CAM module. However, none of the geometric entities created in such tools are parametric, which are not suitable for design changes. They are created using points, lines, and patches, as well as basic extrusion, sweep, and rotate operations. Modeling capabilities in CAM-centered software are in general inferior to those in CAD.

11.5.2 SPECIAL-PURPOSE MACHINING SOFTWARE

Special-purpose software offers toolpath generation for narrow tasks. Some of these codes include ArtCAM and DeskCNC, among many others. These software tools are very much tailored for specific machining tasks in narrow applications.

ArtCAM supports sign makers, woodworkers, engravers, and jewelers who design and manufacture 3D or 2D products from their artwork. Users can create highly intricate personalized or custom 3D models from 2D sketches or photographs. The software guides users through the entire process, from conceptual sketch to the finished piece or mold. An example of “The Black Bull” sign-making in ArtCAM is shown in Figure 11.32(a) for reference.

DeskCNC is another popular code for engraving. The tool supports users to bring in drawings, photographs, or any other computer graphic image and convert it into a DXF file or into G-Code ready to be machined. Screen captures in Figure 11.32(b) show the process of converting an image into the actual toolpath used in creating the finished engraving. The original file is loaded into DeskCNC and converted and/or processed into a grayscale image by numerous available graphics filters. Once the image is processed, DeskCNC converts the picture files directly into toolpaths for an engraving machine.

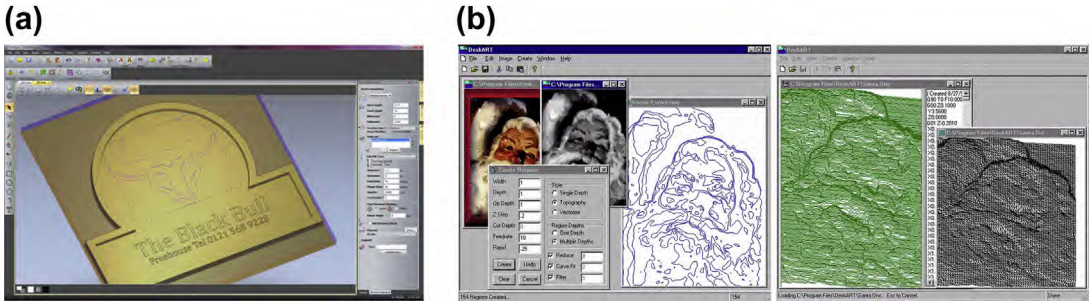


FIGURE 11.32

Examples of special-purpose software: (a) "The Black Bull" sign making in ArtCAM (www.artcam.com/videos/2011-online-demo-catchup.html), and (b) the process of converting an image into the actual toolpaths using DeskCNC (www.deskam.com/deskart.html).

Another special-purpose code worth mentioning is CNCezPRO. It is a real-time 3D simulator for CNC toolpath verification, equipped with good real-time rendering and 3D visualization of tool cuts. Users are able to write, edit, and revise G- and M-codes; and visualize the toolpath in real time. For example, a trajectory milling (or slot milling) on *XZ* (G18) and *YZ* (G19) planes can be generated and simulated in CNCezPRO, as shown in Figure 11.33(a). In addition, a user-defined post-processor for customized G- and M-codes and a macro language for custom cycles, make CNCezPRO a powerful tool-proving package for any shop. Other similar software includes CNC Simulator

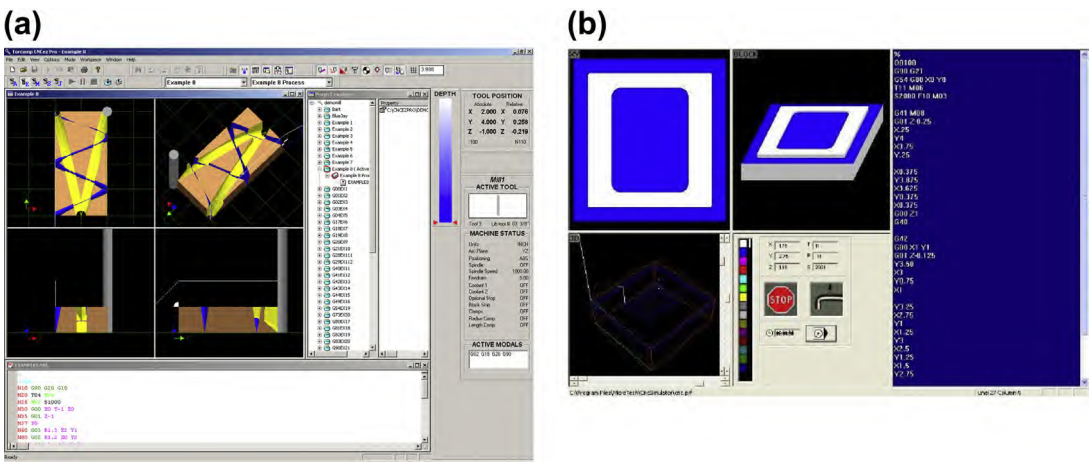


FIGURE 11.33

Examples of special-purpose software for manual CNC programming and toolpath verification: (a) a trajectory milling in CNCezPRO (www.cncezpro.com), and (b) pocket and profile milling using CNC Simulator (www.cncsimulator.com).

(see Figure 11.33(b)), CutViewer (www.cutviewer.com), AutoEditNC (www.betatechnical.com/autonc.htm), and CIMCO Edit (www.cimco.com).

11.6 CASE STUDY AND TUTORIAL EXAMPLES

In this section, we will present a case study where virtual machining is conducted for machining dies that cast tracked vehicle roadarms. In addition to virtual machining, this case involves mold design using Pro/MOLD, a software module of Pro/ENGINEER that supports mold design in a computer. The purpose of presenting this case study is to demonstrate that virtual machining is applicable to practical engineering applications in industry. In addition, tutorial examples, including a name plate and sculpture surface machining, will be discussed. Step-by-step instructions for creating these tutorial examples are given in tutorial M4, S4, and P4 using Mastercam, CAMWorks, and Pro/MFG, respectively.

11.6.1 CASE STUDY

Virtual manufacturing is applied to develop and simulate the manufacturing process for tooling that cast tracked vehicle roadarms. The die-casting process is assumed for fabricating green parts of the roadarm, which involves design and machining of the cover and ejector dies. The machined dies can then be used to die cast the green parts of the roadarm. Most die castings are made from non-ferrous metals, specifically zinc, copper, aluminum, magnesium, lead, pewter, and tin-based alloys. We assume that the roadarms are made from aluminum, which is lightweight with high strength. The schematic illustration of the manufacturing process is given in Figure 11.34. As shown in the figure, die casting produces two parts per operation in this design. Volume milling and drilling are used to machine the green part for the final roadarms by removing extra material that remains in the sprue and runners, as well as drilling holes at each end.

Two dies, cover and ejector, are designed using Pro/MOLD. A sprue, which is a countersink hole, is created on the top face of the cover die, as shown in Figure 11.35(a). Circular shape runners are created between the cover and ejector dies. In addition, waterlines are created in both dies for cooling purposes, as shown in Figure 11.35(a).

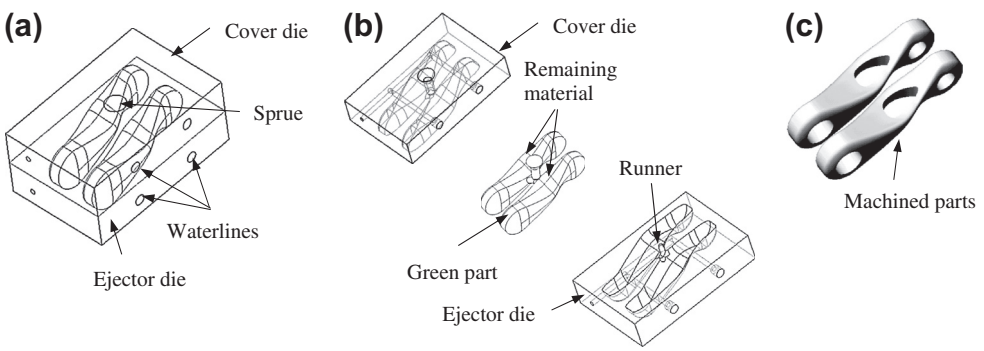


FIGURE 11.34

Schematic illustration of roadarm manufacturing process: (a) mold design, (b) explode view, and (c) machined parts.

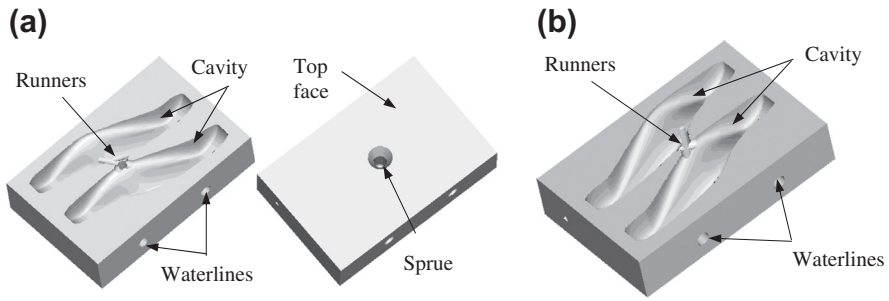


FIGURE 11.35

Design of cover and ejector dies: (a) cover die and (b) ejector die.

Table 11.3 Cutters Used for Volume Milling

	NC 1	NC 2
Length	3 in.	3 in.
Diameter	1 in.	0.25 in.
Round	0.1 in.	0.1 in.

Table 11.4 Machining Parameters for Volume Milling

	NC 1	NC 2
Cut feed	10 in./min.	10 in./min.
Step depth	0.5 in.	0.1 in.
Stepover	0.7 in.	0.2 in.
Spindle speed	2000 rpm	2000 rpm
Machining time	9.2 minutes	58.7 minutes

The machining operations of the dies involve volume milling that removes most of the cavity material (plus sprue and runners), local milling that cleans up material left from the volume milling, contour surface milling that further cleans up the cavity surface, grinding that polishes the cavity surface by removing the leftover scallops, and hole drilling for the waterlines. Hot work H13 die steel is selected as the material for the die workpiece. For illustration purposes, only volume milling and contour surface milling performed on the cover die are shown. To begin the machining operation for the cover die, a workpiece of 8 in. $\times$ 5.25 in. $\times$ 2 in. rectangular block is first created.

A 3-axis mill is assumed for performing two NC sequences of volume milling. The first sequence removes the top thin layer of the workpiece to generate a smooth surface. The thin layer thickness is 0.125 in. The second volume milling cuts the cavity and runners. Cutters and machining parameters of

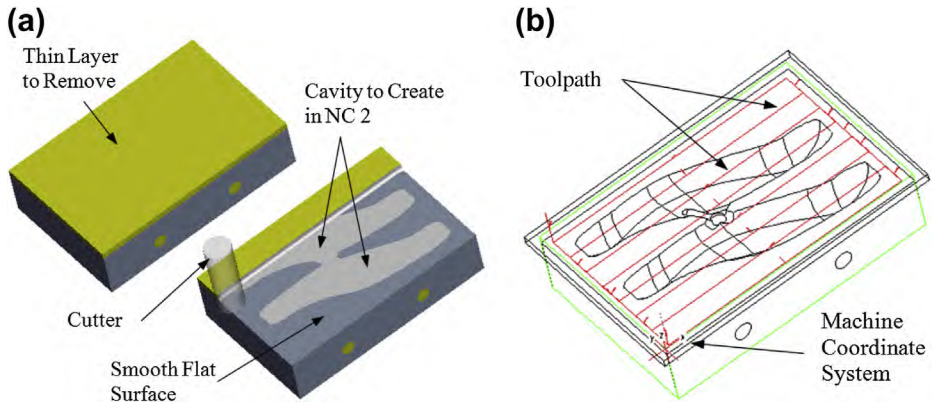


FIGURE 11.36

NC sequence 1 of volume milling: (a) milling simulation and (b) toolpath.

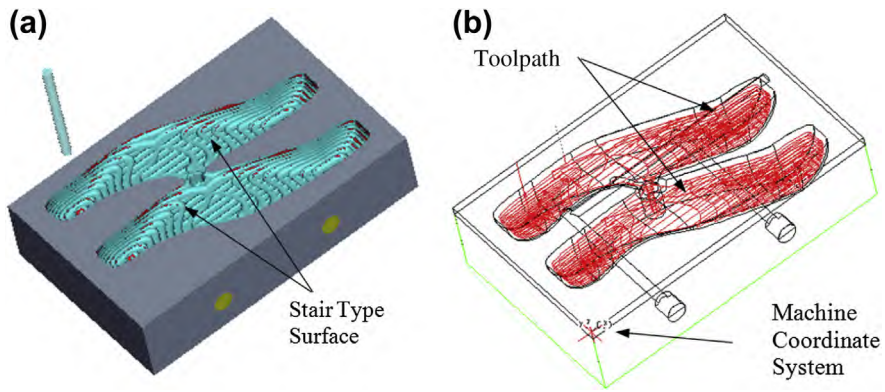


FIGURE 11.37

NC sequence 2 of volume milling: (a) milling simulation and (b) toolpath.

the NC sequences are listed in Tables 11.3 and 11.4, respectively. As shown in Table 11.4, the machining times of the two NC sequences are 9.2 and 58.7 minutes, respectively.

Simulation of these two NC sequences in Pro/MFG is illustrated in Figures 11.36 and 11.37, respectively. The first sequence creates a smooth flat surface. The second sequence cuts the cavity but generates a staircase-type surface, which is not smooth. Note that the distance between the stair layers is controlled by the step depth, which is 0.1 in. in this case. The cavity surface must be machined again using contour surface milling for a smoother surface. This will minimize the grinding operation and improve accuracy.

A 3-axis mill is assumed for performing the contour surface milling. Note that five-axis mill usually produces better surface smoothness for contour surface milling. The cutter of length 2 in.,

Table 11.5 Machining Parameters for Contour Surface Milling

	Contour Surface Milling
Cut feed	10 in./min.
Stepover	0.25 in.
Spindle speed	2000 rpm
Machining time	35.9 minutes

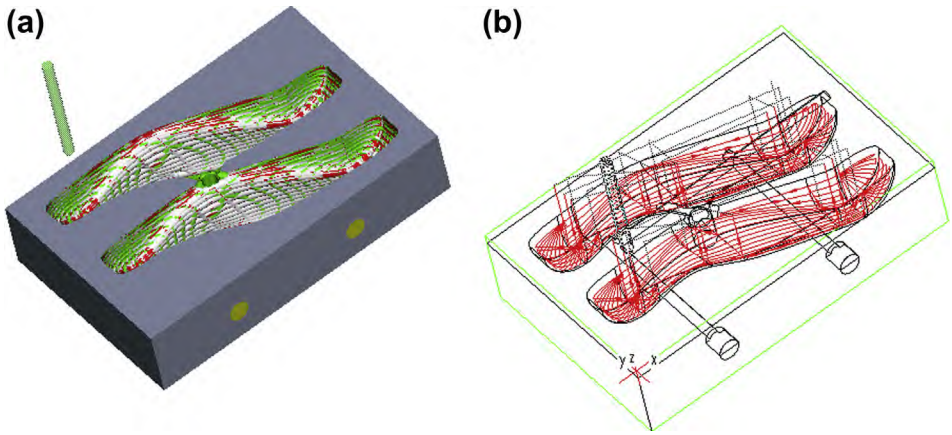


FIGURE 11.38

NC sequence of contour surface milling: (a) milling simulation and (b) toolpath.

diameter 0.25 in., and round radius 0.125 in. is used for the contour milling. The machining parameters are summarized in Table 11.5.

The contour surface milling simulation is performed, as shown in Figure 11.38(a). The toolpath of the machining operation is shown in Figure 11.38(b). The total machining time is estimated as 35.9 minutes. Note that the cavity surface is still not smooth. The scallop remaining on the cavity surface can be reduced by using a smaller stepover and manufacturing time will increase. Grinding operations are often employed to polish the cavity surface, which is not simulated.

11.6.1.1 Virtual Machining for Green Part

With the dies fabricated, the die-casting process can be carried out. Assuming the casting process has been performed, the roadarm green part shown in Figure 11.34(b) is obtained. The volume milling is performed first to remove the extra material remaining in the runners and sprue, then to create holes in the roadarm. Using the roadarm green part as the workpiece, the 3-axis milling machine is again selected.

The volume milling simulation for the center hole is presented, as shown in Figure 11.39(a). The machining time for the volume milling is found to be 29 minutes, and the toolpath is shown in Figure 11.39(b).

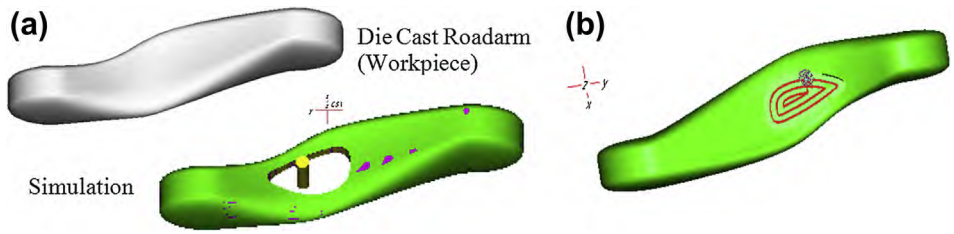


FIGURE 11.39

Volume milling operation for the middle hole: (a) milling simulation and (b) toolpath.

Using the preceding virtual manufacturing process, the manufacturability of the optimal roadarm can be verified. In addition to the simulation and machining time estimate, Pro/MFG generates cutter location data that can be converted into G-code and M-code for CNC using a proper post-processor. Certainly, special fixtures must be design and fabricated to support the machining sequences of the green part and cutting the hole of the roadarm.

11.6.2 TUTORIAL EXAMPLES

A name plate and a sculpture surface example are included in the tutorial lessons. This name plate example involves profile and volume millings and is presented so that you may learn to use Pro/MFG and/or Mastercam in projects P4 and M4, respectively. In addition, after going through this simple example, you should see the major differences between the two software tools, and hopefully be able to identify a few pros and cons of each. Once you are more familiar with one of these two software tools, you may move to the second tutorial example—block with a sculpture surface—in which both rough and finish cuts are required. Three CAM software, Pro/MFG, Mastercam, and CAMWorks are employed for this example. Since some information about this example has been discussed in Section 11.3 using Pro/MFG, we will focus more on Mastercam. Machining the block using CAMWorks is discussed in Tutorial S4.4.5. This example should help you become more familiar with Pro/MFG, Mastercam and/or CAMWorks.

11.6.2.1 Name Plate

This tutorial example involves designing and creating a toolpath for machining a customized name plate on a 3 in. × 12 in. × ⅜ in. workpiece, using both Pro/MFG and Mastercam.

Two NC sequences, volume milling and engraving, are created using Pro/MFG. Nine letters, "Name Plate," are sketched on the top surface of the rectangular block and extruded ⅛ in. into the block as a cut feature, as shown in Figure 11.40(a). Note that a *CG Triumvirate Inserat* font is chosen for the alphabets. You may draw a spline curve and choose "*Place Along Curve*" to place the text along the curve (see Figure 11.40(b)).

An ⅛ in. end mill is picked to make the cut. As shown in the machining simulation, this small cutter is able to reach most pocket areas to make cuts, except the narrow area (e.g., the narrow cross line in the letter "N") and sharp corners. In general, the toolpath generated for volume milling is acceptable.

Note that in creating a volume milling sequence using Pro/MFG, users will have to create a "mill volume," which defines the portion of the workpiece to be removed. In using Pro/MFG, creating a mill

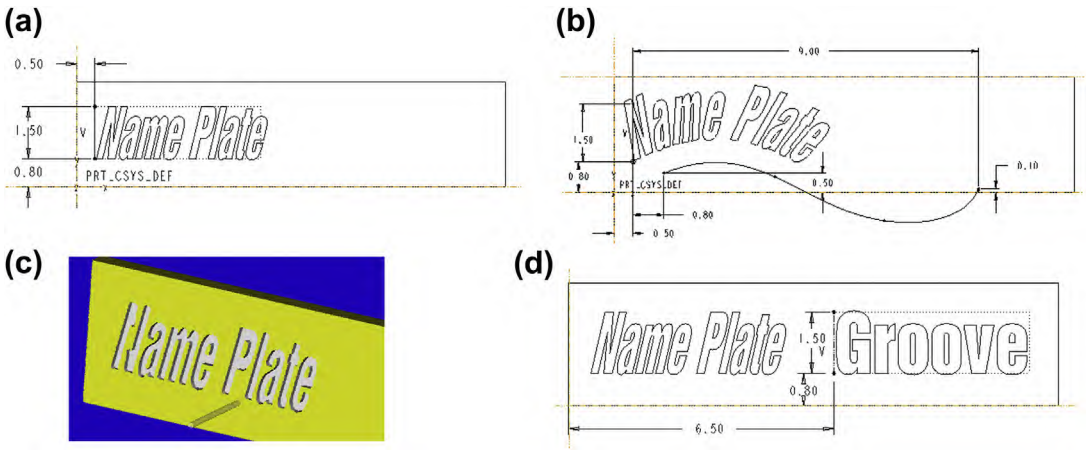


FIGURE 11.40

CNC sequences for machining a name plate using Pro/MFG: (a) creating text for volume milling, (b) adding a spline, (c) machining simulation in Vericut, and (d) adding text for an engraving sequence.

volume involves making a sketch, extruding the sketch for a solid block, and intersecting the block with the design model to generate a trim solid feature that represents exactly the material to be removed. For some intricate surfaces, Pro/MFG may not be able to process the trim function.

The toolpath for the engraving sequence in this example is created using a *Groove* feature in Pro/ENGINEER. As shown in Figure 11.40(d), six letters—"Groove"—in an outline style are defined in the design model. The same cutter with a step depth, ⅛ in., is chosen for the engraving NC sequence. Note that the text engraved will have the same size as the cutter (⅛ in.); that is, the cutter will move one pass along the outer profile of individual letters.

A similar name plate design is created for Mastercam, as shown in Figure 11.41(a). The *MC9* font is chosen to create the "MASTERCAM" text. Note that a simple 2D wireframe sketch is sufficient for this machining simulation; there is no need to create 3D solid models. Two sequences are created, a profile milling that cuts along the outer boundary of the workpiece, and a volume milling that cuts the area between the inner boundary curve and the outer profile of individual letters. An ⅛ in. end mill with a step depth of ⅛ in. is picked to make the cut. The toolpath (profile milling and volume milling combined) and machining simulation are shown in Figures 11.41(b) and (c), respectively.

11.6.2.2 Block with a Sculpture Surface

NC sequences for machining the sculpture surface of the block example discussed in Section 11.3 are chosen as the second tutorial example. As discussed in Section 11.3, three NC sequences—volume milling, local milling, and contour surface milling—were employed for machining the surface using Pro/MFG. This tutorial focuses on using Mastercam to create a similar machining simulation.

We will create three sequences using Mastercam—surface rough flowline, rough-restmill, and surface finish—which correspond to volume milling, local milling, and contour surface milling in Pro/MFG. Since creating the sculpture surface by blending circular arcs on four parallel planes using Mastercam is more time consuming, we will import the Pro/ENGINEER part of the block into

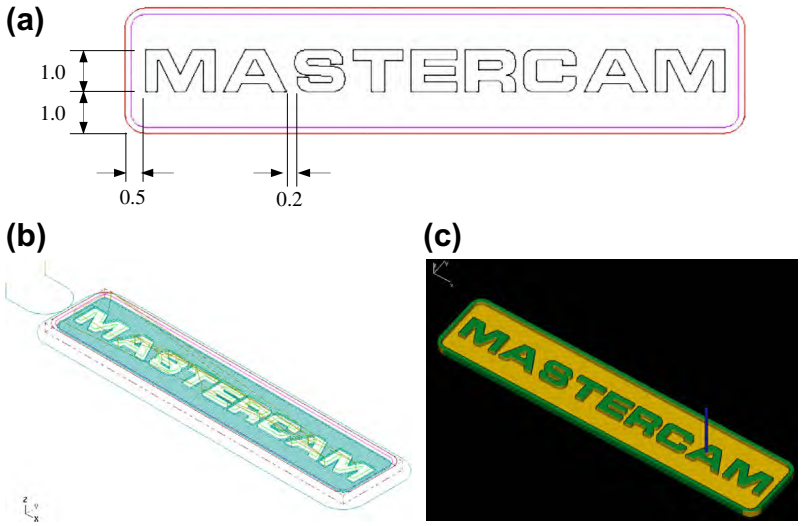


FIGURE 11.41

Volume CNC sequences for machining a name plate using Mastercam: (a) creating text for volume milling, (b) combined toolpath, and (c) machining simulation.

Mastercam. Mastercam supports model import directly from Pro/ENGINEER in addition to other CAD systems (e.g., SolidWorks, CATIA, and AutoDesk).

The Pro/ENGINEER part will be imported into Mastercam in a default position and orientation, similar to that of Figure 11.42(a). The imported geometric entities usually must be rotated and translated to a proper position and orientation (similar to those shown in Figures 11.42(b) and (c)) so that the machine zero coincides and aligns with the coordinate system built in Mastercam.

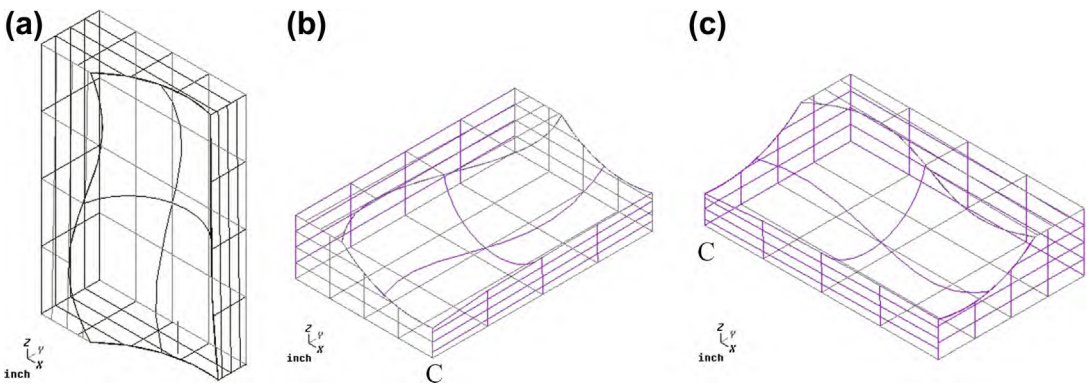


FIGURE 11.42

Adjusting position and orientation for the imported part: (a) default position and orientation, (b) rotate −90° along *X*-axis, and (c) rotate 90° along *Z*-axis, and then translate 4 units along the *Z*-axis.

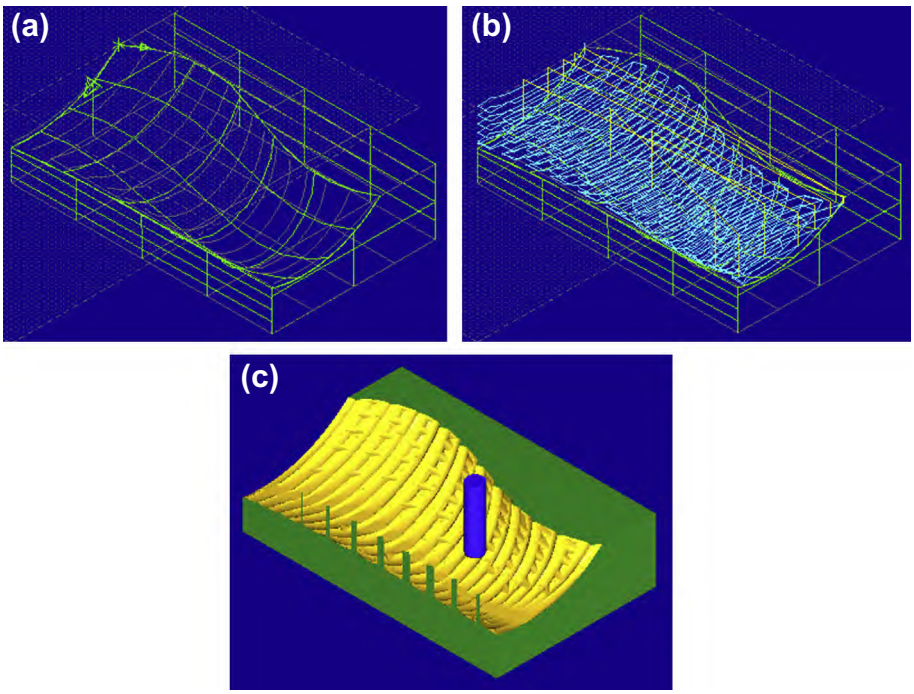


FIGURE 11.43

Surface rough flowline machining in Mastercam; (a) flowline direction, (b) toolpath, and (c) machining simulation.

A surface rough flowline with a 1 in. bull mill of ¼ in. corner radius was chosen for the rough cut. Flowline is one of the several options available in Mastercam. Flowline cuts single or multiple surfaces using their natural shape to define the toolpath. Stepover and step depth are 0.4 in. and 0.2 in., respectively. These parameters are identical to those defined in Pro/MFG. A flowline direction along the short edge of the surface (Figure 11.43(a)) is chosen. The resulting toolpath and machining simulation are shown in Figures 11.43(b) and (c), respectively.

As can be seen in Figure 11.43(b), Rough Flowline does a similar cut to that of volume milling in Pro/MFG. Mastercam does not require defining mill volume as that of Pro/MFG and requires fewer steps to generate a similar toolpath.

The next sequence is Rough-Restmill, which removes remaining materials from the previous rough cut. Rough-Restmill is very similar to local milling in Pro/MFG. The same ½ in. ball-nose cutter and the same stepover and maximum step depth are chosen to perform this sequence. The resulting toolpath and machining simulation are shown in Figures 11.44(a) and (b), respectively. Note that the restmill toolpath follows a spiral pattern similar to that of Pro/MFG.

The third and final sequence is finish cut. A finish-scallop and the same ½ in. ball-nose cutter are chosen. The finish-scallop option generates finishing toolpaths that have consistent scallop heights over the entire set of surfaces, which is a desired option to perform a finish cut. A maximum stepover, 0.05 in., is specified for this sequence. Note that a similar option is not available in Pro/MFG. The resulting

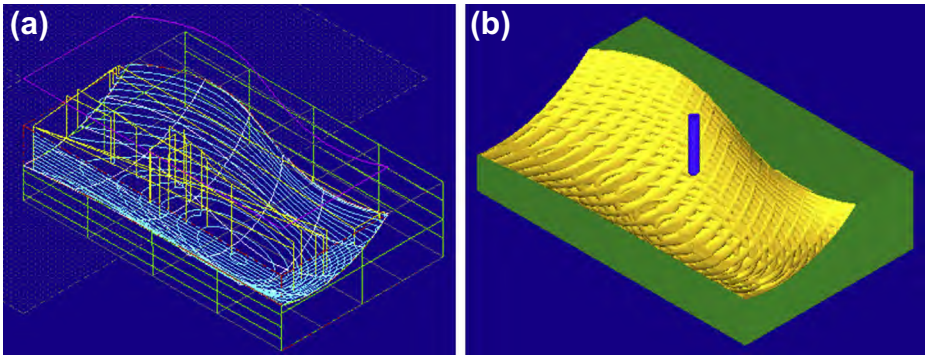


FIGURE 11.44

Surface rough restmill machining in Mastercam: (a) toolpath and (b) machining simulation.

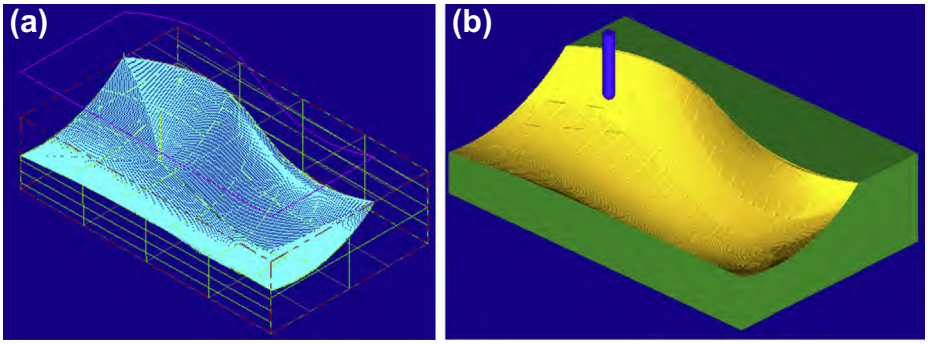


FIGURE 11.45

Surface finish-scallop machining in Mastercam: (a) toolpath and (b) machining simulation.

toolpath and machining simulation are shown in Figures 11.45(a) and (b), respectively. As shown in Figure 11.45(b), the surface looks fairly smooth. This is because a very small maximum stepover (0.05 in.) is specified for the finish cut. As a result, the overall machining time estimated is 7 hours and 18 minutes, as reported by Mastercam.

11.7 SUMMARY

In this chapter, we discussed the subject of virtual machining; including NC part programming, virtual machining simulations, simulation software that is commercially available, as well as a case study and tutorial examples. In addition, we briefly addressed the practical aspects in CNC machining, including jigs and fixtures, and cutters and machining parameters. We hope this chapter has helped you become

more familiar with the subject. With more practice, you should become more confident and competent in using virtual machining simulation tools to address the manufacturability of the product and obtain reasonable machining time estimates to support product design.

In the next chapter, we will introduce the topic of toolpath generation, which is the backbone of virtual machining simulation. It should provide you with a more in-depth understanding on the factors that contribute to the CNC toolpath generated in virtual machining simulation. For those who are conducting research relevant to virtual machining or toolpath generation, Chapter 12 should serve well as an introductory article for understanding toolpath generation. For practicing engineers, the chapter offers behind-the-scene operations in the calculation of cutter location data and parameters that affect that data. Knowledge acquired will help you diagnose problems encountered on the machined surfaces (e.g., gouging).

In the remaining chapters of Part III′ Product Manufacturing and Cost Estimating, we will describe other pertinent technology that supports addressing the manufacturing issues from design perspectives, including sheet forming simulation (Chapter 13), rapid prototyping (Chapter 14), and manufacturing cost estimates (Chapter 15). CNC machining discussed in this chapter and rapid prototyping in Chapter 14 offer a means to support fast prototyping in product development. These chapters should provide you with a very good understanding on the essential topics involved in product manufacturing using advanced computer-aided technology.

APPENDIX 11A: SAMPLE ADDRESS CODES

A	Fourth-axis rotary motion	The A address character is used to specify motion for the optional fourth, A, axis. It specifies an angle in degrees for the rotary axis. It is always followed by a signed number and up to three fractional decimal positions
B	Fifth-axis rotary motion	The B address character is used to specify motion for the optional fifth, B, axis. It specifies an angle in degrees or the rotary axis. It is always followed by a signed number and up to three fractional decimal positions
C	Auxiliary external rotary axis	The C address character is used to specify motion for the optional external sixth, C, axis. It specifies an angle in degrees for the rotary axis. It is always followed by a signed number and up to three fractional decimal positions
D	Tool diameter selection	The D address character is used to select the tool diameter or radius used for cutter compensation
F	Feedrate	The F address character is used to select the feedrate applied to any interpolation functions, including pocket milling and canned cycles
G	Preparatory functions (G-codes)	The G address character is used to specify the type of operation to occur in a block
H	Tool length offset selection	The H address character is used to select the tool length offset entry from the offset's memory

Continued

I, J, K	Canned cycle and circular optional data	The I, J, and K address character is used to specify data used for some canned cycles and circular motions
L	Loop count for repeated cycles	The L address character is used to specify a repetition count for some canned cycles and auxiliary functions
M	Code miscellaneous functions	The M address character is used to specify an M-code for a block. These codes are used to control miscellaneous machine functions
N	Number of block	The N address character is entirely optional. It can be used to identify or number each block of a program
R	Canned cycle and circular optional data	The R address character is used in canned cycles and circular interpolation
S	Spindle speed command	The S address character is used to specify the spindle speed
T	Tool selection code	The T address character is used to select the tool for the next tool change
X, *Y*, *Z*	Linear *X*-, *Y*-, and *Z*-axis motions	The *X*-, *Y*-, and *Z*-address characters are used to specify motion for the *X*-axis. It specifies a position or distance along the *X*-, *Y*-, and *Z*-axes, respectively

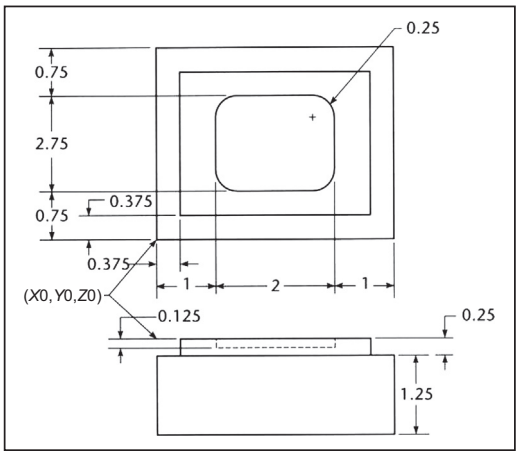
APPENDIX 11B: SAMPLE G- AND M-CODES

G-Code	Function
G00	Rapid Motion
G01	Linear Interpolation Motion
G02	CW Interpolation Motion
G03	CCW Interpolation Motion
G04	Dwell
G12	CW Circular Pocket Milling
G13	CCW Circular Pocket Milling
G17	*XY* Plane Selection
G18	*ZX* Plane Selection
G19	*YZ* Plane Selection
G20	Select Inches
G21	Select Metric
G28	Return to Reference Point
G29	Return From Reference Point
G40	Cutter Comp Cancel
G41	2D Cutter Compensation Left
G42	2D Cutter Compensation Right
G43	Tool Length Compensation +

G44	Tool Length Compensation –
G47	Text Engraving
G49	G43/G44/G143 Cancel
G80	Canned Cycle Cancel
G81	Drill Canned Cycle
G82	Spot Drill Canned Cycle
G83	Normal Peck Drill Canned Cycle
G84	Tapping Canned Cycle
G85	Boring Canned Cycle
G86	Bore/Stop Canned Cycle
G87	Bore/Stop/Manual Retract Canned Cycle
G88	Bore/Dwell/Manual Retract Canned Cycle
G89	Bore/Dwell Canned Cycle
G90	Absolute
G91	Incremental
G98	Initial Point Return
G99	R Plane Return
M-Code	**Function**
M00	Stop Program
M01	Optional Program Stop
M02	Program End
M03	Spindle Forward
M04	Spindle Reverse
M05	Spindle Stop
M06	Tool Change
M08	Coolant On
M09	Coolant Off
M30	Prog End and Rewind
M97	Local Sub-Program Call
M98	Sub Program Call
M99	Sub Program Return or Loop

QUESTIONS AND EXERCISES

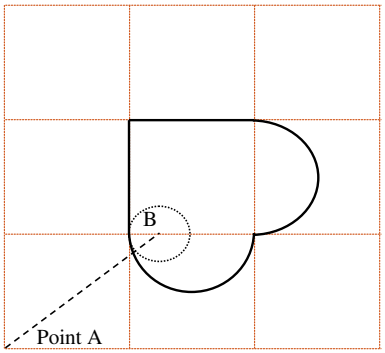
11.1. Write a CNC program (with explanations for each block) for a HAAS mill to machine a 4 in. × 4.25 in. × 2.5 in. aluminum block for the part (design model) shown in the following image and table. Choose your own cutters (from those shown in the table) and cutting parameters (step depth and stepover) with "common sense." Note that you need to use tool radius compensation for all toolpaths. Calculate the overall cutting time.



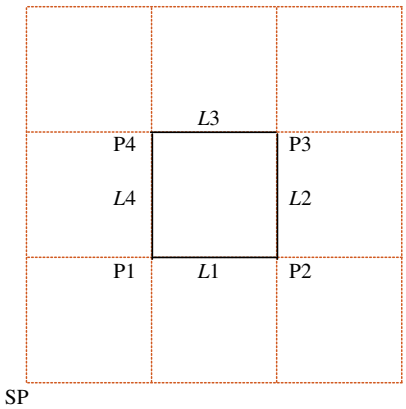
Size (inches)	Type	Length (inches)
1/8	Ball-nose	0.375
3/16	Ball-nose	1.000
1/4	Ball-nose	0.625
1/2	Ball-nose	2.000
1/8	Flat-end	0.375
3/16	Flat-end	0.475
1/4	Flat-end	0.500
1/4	Flat-end	0.875
5/16	Flat-end	0.600
3/8	Flat-end	0.600
1/2	Flat-end	1.000
1/2	Flat-end	1.500

11.2. Verify your NC program of Problem 1 using an adequate software tool. Search and find a software program that takes your NC program and simulates the NC sequence.

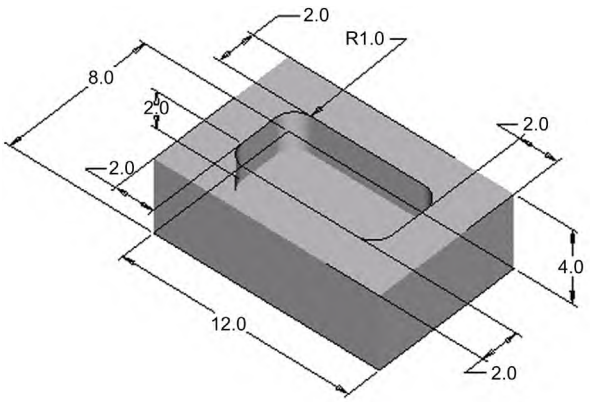
11.3. Write an NC program to cut a 0.25 in.-deep pocket on a 3 in. × 3 in. × 0.5 in. block shown in the following figure. The cutter diameter is 0.5 in. Note that the cutter will cut inside of the pocket by moving along the pocket boundary **only one time**, which will leave some material uncut. You must use tool radius compensation for the NC codes. Identify the uncut area. Also, sketch your toolpath, which must start from point A.

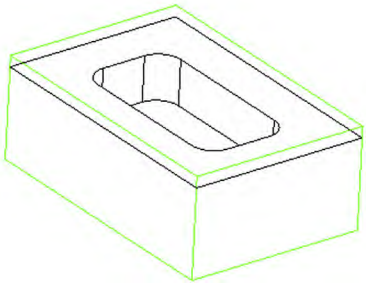


11.4. Write a complete APT (Automatically Programmed Tools) program to cut a 0.25 in.-deep pocket on a 3 in. × 3 in. × 0.5 in. block shown in the following figure. The cutter diameter is 0.75 in. Note that spindle speed is 580 and feedrate is 2.3. Note that the cutter must start from point SP and return to SP after the cut.



11.5. Use Pro/MFG, CAMWorks, or Mastercam (or other CAM software you have access) to machine the design model shown below (left) from a workpiece of 12 in.×8 in.×4.5 in. (show below right). Note that at least two NC sequences are needed: face milling that removes top layer of 0.5 in. deep from top surface of the workpiece, then pocket milling that cuts the pocket.





11.6. Design and machine a nameplate on a 3 in.×12 in. × ⅜ in. Plexiglas workpiece using the HAAS CNC mill (or other mill you have access to). Note that you can design your plate using Pro/MFG (CAMWorks or Mastercam), directly write your CNC codes using HAAS commands, or any other software tools that you found. This will be a combination of text engraving, volume and trajectory milling, and picture/logo carving. The following is what you need to submit for grade. Please BE VERY CREATIVE.

- **a.** Your design, describe it briefly in a few sentences and use pictures for illustration;
- **b.** Your CNC codes with explanations (only important blocks, no need to submit the entire NC program);

c. Final machined nameplate (if applicable);
d. Two significant things that you learned from using the mill.

REFERENCES

Chang, T.-C., Wysk, R.A., Wang, H.-P., 1998. Computer-Aided Manufacturing, second ed. Prentice Hall.
Kalpakjian, S., Schmid, S.R., 2010. Manufacturing, Engineering and Technology, sixth ed. Prentice Hall.
Krar, S., Gill, A., Smid, P., 2010. Technology of Machine Tools, seventh ed. McGraw-Hill.
Oberg, E., Jones, F., Horton, H., Ryffel, H., McCauley, C., 2012. Machinery's Handbook, twentyninth ed. Industrial Press.
Sarma, R., Dutta, D., 1997. An integrated system for NC machining of multi-patch surfaces. Computer-Aided Design 29 (11), 741–749.
Smid, P., 2000. CNC Programming Handbook, second ed. Industrial Press Inc, 200 Madison Avenue, New York, NY. 10016-4078.

CHAPTER

TOOLPATH GENERATION 12

CHAPTER OUTLINE

12.1 Introduction 648
12.2 Inclined Flat Surface 650
12.3 Ruled Surface 657
12.3.1 5-Axis Mill with Ball-Nose Cutter (OP010) 658
12.3.1.1 Number of Passes 658
12.3.1.2 Scallop Height 659
12.3.1.3 Parametric Surface and CL Data 662
12.3.1.4 A Few Questions 666
12.3.2 3-Axis Mill with Flat-End Cutter (OP030) 668
12.3.2.1 Flat-End Cutter 668
12.3.3 3-Axis Mill with Ball-Nose Cutter (OP030) 671
12.3.3.1 CL Data 671
12.3.4 4-Axis Mill with Flat-End Cutter* (OP020) 674
12.3.4.1 CL Data 675
12.4 Cylindrical Surface of Bézier Curve 677
12.5 Summary 680
Questions and Exercises 680
References 683

Toolpath generation is the backbone of virtual machining. Learning basic operations of the virtual machining software and practical procedures in setting up and running a CNC machine is important for design engineers to gain valuable experience in manufacturing. Engineers with the knowledge and experience in virtual machining and CNC operations will naturally be able to incorporate manufacturing issues into design considerations during product development.

In this chapter, we will move one step deeper in virtual machining by discussing toolpath generation—that is, the computations that create cutter location (CL) data. These are behind-the-scene operations when users click on menu buttons in virtual machining software to launch toolpath computation modules. Understanding toolpath computations, the influence of manufacturing parameters (e.g., scallop height or stepover), and the selection of cutters and work cells (e.g., 3- vs. 5-axis mill) to the toolpath and quality of the machined surface will become clear. This chapter should help you become more familiar with toolpath generation and become more confident while selecting cutters and machining parameters in performing virtual and CNC machining.

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00012-0

Copyright © 2015 Elsevier Inc. All rights reserved.

The chapter's focus is on toolpath generation for surface milling since that is much more involved, especially on contour surface milling where requirements of surface finish largely determine the adequacy of the toolpath. An adequate toolpath must create a machined surface with a desired surface finish (governed by scallop height and tolerance) and accuracy, and requires a minimum machining time (or minimum tool movement). Surface milling will be discussed in this chapter.

Toolpath generation for surface milling is a broad topic. Many methods have been proposed by researchers over the decades. Some of these promising methods are adopted and implemented in commercial CAM software. Instead of providing a thorough and in-depth review on various methods for toolpath generation and comparing their implementation in multiple virtual machining software, we will narrow the scope by focusing only on the classical toolpath generation methods, which are intuitive and predicable without hidden recipe or patching. Such classical methods have been implemented in Pro/MFG. Other CAM tools, such as Mastercam, add recipes and patches to these basic methods in offering users flexibility and in some cases better quality in toolpath generation. Discussing these additions may divert our focus. Therefore, examples employed for illustration are mostly created in Pro/MFG. Toolpath calculations are checked with those created in Pro/MFG. Again, this is mainly because the toolpath generation implemented in Pro/MFG is the most basic, intuitive, and predicable; since it follows classical and clean algorithms without any hidden recipe or patching. With access to Pro/MFG, you should be able to take full advantage of this chapter. Even without access to Pro/MFG, we offer you adequate insight into the toolpath generation for surface milling; this should help you understand the CAM software to which you have access.

Computer files, including part and manufacturing models, CL data files, and spreadsheets for calculations, are provided. For those who do not have access to Pro/MFG, you can create the same surface milling sequences using the virtual machining software available to you in order to go through this chapter. If not, you may just use the CL data files and spreadsheets provided to gain as much understanding as possible from this chapter.

What is in this chapter follows a project-based approach. We will start with a very simple case; that is, an inclined flat surface in which the toolpath is very easy to figure out without sophisticated toolpath generation algorithms. We will then discuss two more projects: a ruled surface and a cylindrical surface of a Bézier curve. In each case, we will also discuss the scallop height calculation under various scenarios. The following are the overall objectives of this chapter to provide a general understanding on toolpath generation, specifically for surface milling; to help you understand the impact of machining parameters and cutters on the resulting toolpath or CL data; to offer a detailed discussion on scallop-height calculations, which determine the quality of the machined surface with a quantitative measure.

12.1 INTRODUCTION

The main objective of toolpath generation is to compute a sequence of cutter location data (CL data) on the design surface, which is the surface to be machined. A toolpath is required for any type of NC sequences in the context of virtual machining. Among the many different NC sequences, toolpath generation for contour surface milling is the most complex and has been investigated for many years. In general, toolpath generation methods for contour surface milling are classified as either the CC-based (cutter contact) method or the CL-based method depending on the type of design surface

(Mishra et al., 2005). Both methods aim at generating toolpaths that produce a required small scallop with a minimum cutting path (therefore, minimum machining time and cost).

In the CC-based method, toolpaths are generated by sampling a sequence of CC points from the design surface and then each CC point is converted to a CL point. In general this method can be classified into three main categories: (1) parametric method, (2) drive surface method, and (3) guide plane method, depending on path planning techniques. This CC-based method has been adopted by Parametric Technology and implemented in Pro/MFG.

The *parametric method* computes the toolpaths, as depicted in Figure 12.1(a), in the parametric domain and then maps it back to the design/part surface. Iso-parametric machining, one of the earliest techniques (Sata et al., 1981), involves machining along iso-parameter lines (or iso-line). Uniformly spaced parametric lines in the parametric domain are nonuniformly spaced on the part surface due to nonuniform transformation between parametric and Euclidean spaces. This results in varying scallop height and thus a nonuniform surface finish.

The *drive surface method* computes toolpaths (see Figure 12.1(b)) by intersecting the design surface with a series of planes called drive surfaces. The curves generated by intersection are then used to generate accurate toolpaths. If the intersecting planes are parallel, then it is known as iso-planar machining. However, the orientation of the intersecting plane can be changed to achieve optimal machining conditions. This method is very robust and can handle complex surfaces reliably.

The *guide plane method* plans the toolpaths (either in the form of a straight line or contours) first on a 2D plane and then maps it back (Figure 12.1(c)) to the design surface. For 3-axis milling, the plane perpendicular to the tool axis is mainly used as the guide plane. The major strength of this method is that the shape of the region to be cut on the part surface can be considered while planning the toolpath layout on the guide plane.

With the CL-based method, the CL surface is used as a path-generation surface. In this approach, the CL surface has to be generated first from the design surface. Surface-finish criteria are met on the

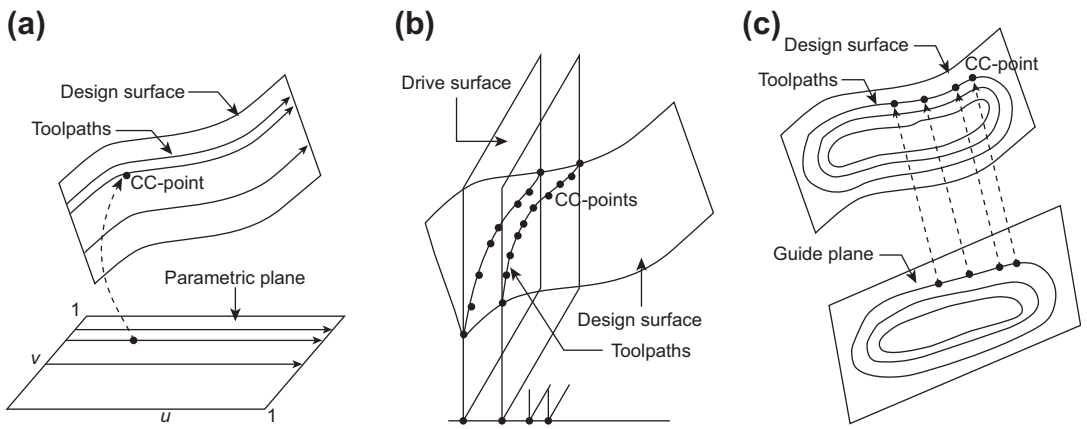


FIGURE 12.1

CC-based toolpath generation methods, based on Sarma and Dutta (1997), Choi and Jerrard (1998), and Kim and Choi (2000): (a) parametric method, (b) drive surface method, and (c) guide plane method.

CL surface. In most of the previous work in this area (Kim and Kim, 1995; Choi et al., 1997), the offset surface was approximated as the CL surface. The offset surface is first generated from the design surface and toolpaths are planned on the offset surface. Zig-zag toolpaths (Misra et al. 2005), parallel to each other, are planned according to the machining parameters.

The objective of this chapter is to provide readers with a practical discussion on how the toolpath is generated in CAM software. With such a discussion, readers should develop better skill and understanding of selecting proper machining parameters and cutters leading to toolpaths that produce the required small scallop with a minimum cutting path. With this in mind, we will narrow the discussion to one toolpath generation method, instead of providing a detailed and thorough presentation on the subject. We will focus on the *parametric method* of the CC-based methods. This is one of the options provided in Pro/MFG for contour surface milling, among other CAM software.

The following presents three examples—inclined flat surface, ruled surface, and cylindrical surface of Bézier curve—to discuss toolpath and scallop height of the machined surface. Toolpaths generated in all three examples are compared with those created in Pro/MFG for verification.

12.2 INCLINED FLAT SURFACE

A flat surface of 45° (10 in. × 5 in.), shown in Figure 12.2(a), is to be machined. A flat-end cutter of 1.2 in. diameter is chosen with a stepover 1.0 in. The machine zero is defined at the top left corner of the workpiece (i.e., coordinate system *CS*1), as shown in the figure. The toolpath generated using a 5-axis mill following the parametric method is shown in Figure 12.2(b). Although the machined surface seems to be clean (Figure 12.2(c)), remember that in practice NC sequences for rough cut (e.g., volume milling) must be carried out prior to the surface milling sequence.

As shown in Figure 12.2(b), six straight lines are created on the design surface. These straight lines are CC lines; they are evenly spaced with spacing between neighboring lines of 1 in., which is the stepover specified. Since a 5-axis mill is employed, the cutter is oriented in a normal direction to the design surface. In this case, the CL points stay on CC lines. In addition, there is no need for the cutter to adjust its orientation while it moves from one end of the tool pass to the other; thus, no intermediate CL

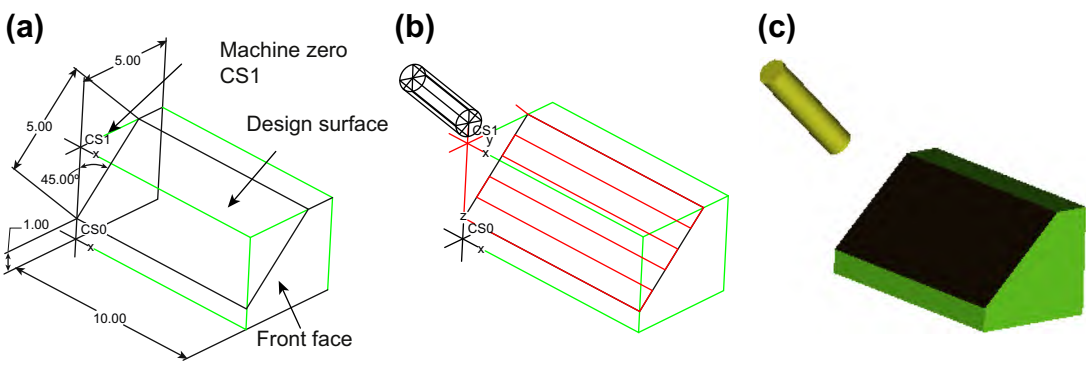
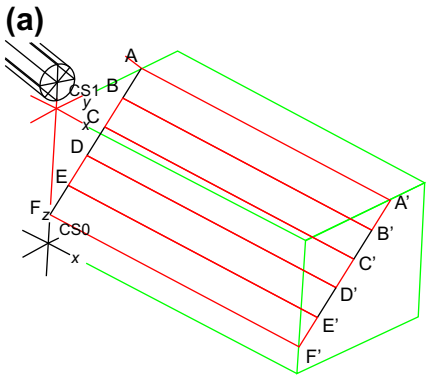


FIGURE 12.2

Contour surface milling for an inclined flat surface: (a) design surface, (b) toolpath generated by Pro/MFG, and (c) machining simulation.



(b)

Point	*x*	*y*	*z*	*nx*	*ny*	*nz*
A	0	3.536	0	0	–0.707	0.707
A'	10	3.536	0	0	–0.707	0.707
B'	10	2.828	–0.707	0	–0.707	0.707
B	0	2.828	–0.707	0	–0.707	0.707
C	0	2.121	–1.414	0	–0.707	0.707
C'	10	2.121	–1.414	0	–0.707	0.707
D'	10	1.414	–2.121	0	–0.707	0.707
D	0	1.414	–2.121	0	–0.707	0.707
E	0	0.707	–2.828	0	–0.707	0.707
E'	10	0.707	–2.828	0	–0.707	0.707
F'	10	0	–3.536	0	–0.707	0.707
F	0	0	–3.536	0	–0.707	0.707

FIGURE 12.3

Toolpath: (a) CC line and CL points and (b) CL data.

points are needed. As a result, the cutter is simply moving on the design surface back and forth following the CC lines.

More specifically, the toolpath is from point A to A′, stepover to B′, and then moves to B, stepover to C, then moves to C′, and so on (Figure 12.3(a)). The CL points can be easily calculated with reference to machine zero *CS*1. The CL data points consist of *X*-, *Y*-, and *Z*-locations, as well as components of the unit normal vector in *X*-, *Y*-, and *Z*-coordinates that define the cutter orientation, as listed in Figure 12.3(b).

Next, we assume a 3-axis mill with the same cutter and stepover. In this case, CC lines are identical to those of the 5-axis case, employing the parametric method. However, CL data points do not stay on the CC lines in this case, as can be seen in Figures 12.4(a) and (b). From a machining simulation shown in Figure 12.4(c), there is a large amount of material remaining on the design surface, which is of a stair-type shape viewed from the side.

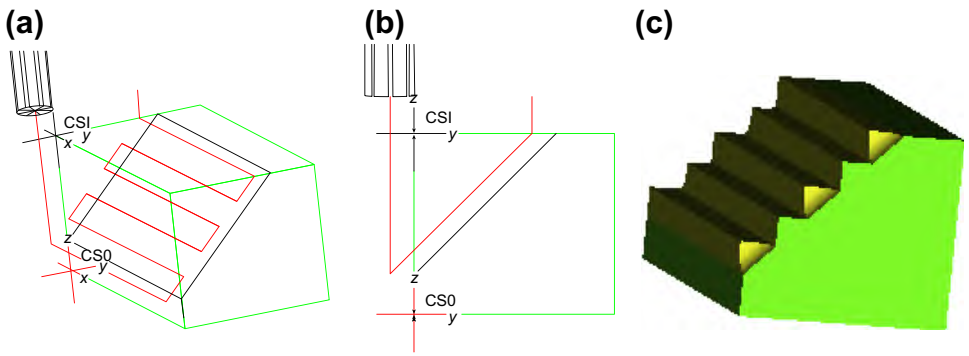
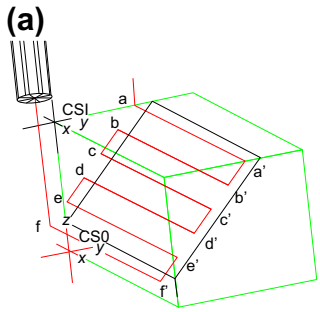
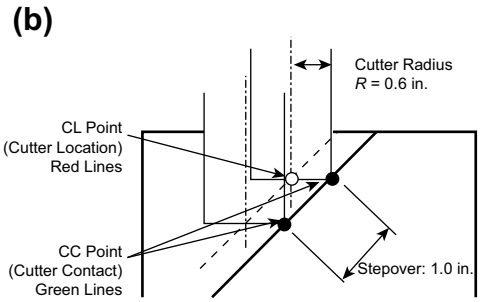


FIGURE 12.4

Contour surface milling using 3-axis mill: (a) toolpath (iso-view), (b) toolpath (side view), and (c) machining simulation.





(c)

Point	x	y	z
a	0	2.936	0
a'	10	2.936	0
b'	10	2.228	–0.707
b	0	2.228	–0.707
c	0	1.521	–1.414
c'	10	1.521	–1.414
d'	10	0.814	–2.121
d	0	0.814	–2.121
e	0	0.107	–2.828
e'	10	0.107	–2.828
f'	10	–0.6	–3.536
f	0	–0.6	–3.536

FIGURE 12.5

Toolpath: (a) CL points, (b) CL points offset from CC, and (c) CL data.

As can be seen in Figure 12.4(b), a CL point is offset horizontal to the left by an amount of the cutter radius. The offset is illustrated more clearly in Figure 12.5(a). As a result, the CL data can be calculated easily by subtracting the cutter radius (0.6 in.) from the *Y*-coordinate of the corresponding points on the CC lines. Note that for a 3-axis mill, the cutter is always aligned vertically with the *Z*-axis; therefore, there is no need to include a normal vector that defines the cutter orientation.

Calculation of scallop height is straightforward in this case. Scallop height is defined as the maximum distance between the design surface and the material remaining on the machined surface. As shown in Figure 12.6, the scallop height h can be calculated by

$$h = (S \cos \gamma)\sin \gamma \tag{12.1}$$

where S is the stepover and γ is the slope angle of the design surface. For this example, γ is 45° and S is 1.0 in.; thus, scallop height h is 0.5 in. Note that by using a flat-end cutter and a 3-axis mill for surface milling, the scallop height is independent of the cutter diameter.

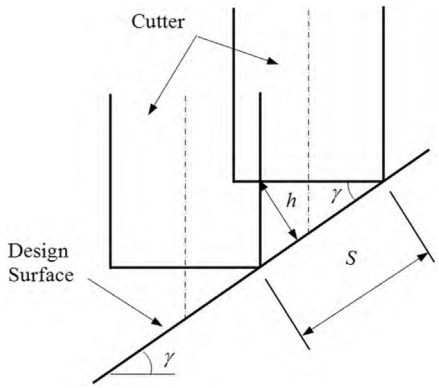


FIGURE 12.6

Scallop height calculation.

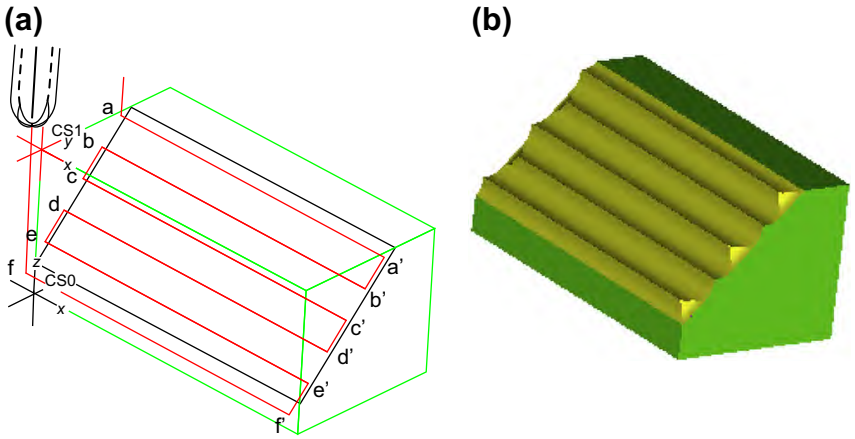
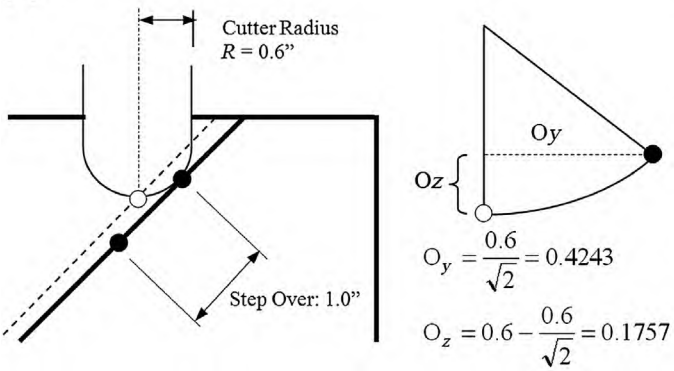


FIGURE 12.7

Contour surface milling using 3-axis mill with ball-nose cutter: (a) toolpath (iso-view), and (b) machining simulation.

If a ball-nose cutter, instead of a flat-end cutter, is employed with a 3-axis mill, the toolpath generated by using the parametric method (also implemented in Pro/MFG) is shown in Figure 12.7(a). Similar to the flat-end cutter, the CL data do not coincide with the CC lines; however, they are closer to them. Again, material remains on the machined surface, which is less than in the flat-end cutter case, as shown in Figure 12.7(b). The key questions to ask are: How are the CL points offset from the CC? How can scallop height be calculated?

(a)



(b)

Point	x	y	z
a	0	3.111	-0.176
a'	10	3.111	-0.176
b'	10	2.404	-0.883
b	0	2.404	-0.883
c	0	1.697	-1.590
c'	10	1.697	-1.590
d'	10	0.899	-2.297
d	0	0.899	-2.297
e	0	0.283	-3.004
e'	10	0.283	-3.004
f'	10	-0.424	-3.711
f	0	-0.424	-3.711

FIGURE 12.8

Toolpath for ball-nose cutter: (a) CL points offset from CC and (b) CL data.

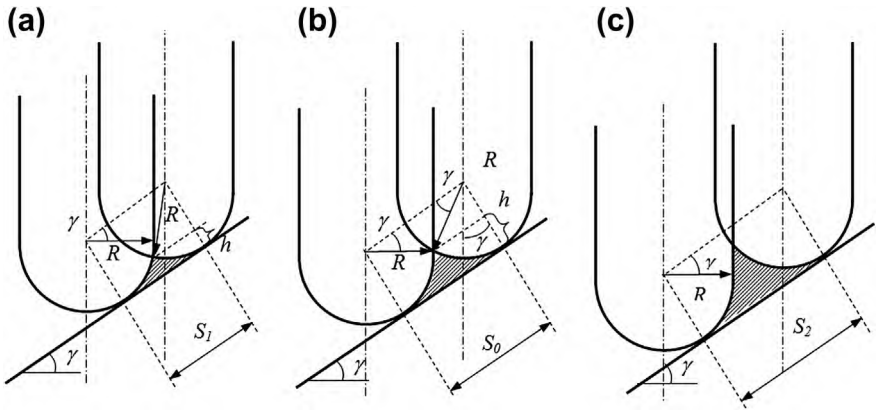


FIGURE 12.9

Scallop-height calculations: (a) Case 1, symmetric circular shape; (b) symmetric circular shape; and (c) Case 2, asymmetric shape.

The CL point is offset in both *Y*- and *Z*-directions, as illustrated in Figure 12.8(a). In general, the offset can be calculated simply from trigonometry, as

$$O_y = R \sin \gamma \tag{12.2a}$$

$$O_z = R(1 - \cos \gamma) \tag{12.2b}$$

where R is the cutter radius. For this example, γ is 45° and R is 0.6 in.; thus, the offset is $O_y = 0.4243$ in. and $O_z = 0.1757$ in. The CL data can then be calculated by using

$$CL_x = CC_x \tag{12.3a}$$

$$CL_y = CC_y - O_y \tag{12.3b}$$

$$CL_z = CC_z - O_z \tag{12.3c}$$

The CL data calculated are listed in Figure 12.8(b), and are identical to those generated by Pro/MFG.

Calculation of scallop height for this NC sequence (i.e., 3-axis mill with a ball-nose cutter) is a little more involved. There are basically two possible cases, depending on the slope of the design surface (γ), the stepover (S), and the cutter diameter. In these two cases, different equations will be formulated to determine scallop heights. Which case to choose will be determined by the condition if the shape of the scallop is symmetric with respect to the line that is perpendicular to the surface and passes through the intersecting point of the cutter silhouette contours between passes, as shown in Figure 12.9. Note that in the figure, we assume $S_1 < S_0 < S_2$ for the given slope and stepover.

Figure 12.9(b) illustrates the condition that separates the two cases, where $R \cos \gamma = S/2$. Therefore, for Case 1: $R \cos \gamma \geq S/2$; and for Case 2: $R \cos \gamma < S/2$. In each case, we will calculate scallop height with a given number of passes. We will also learn how to calculate the number of passes for a prescribed allowable scallop height.

CASE 1: $R \cos \gamma \geq S/2$

First, we will discuss how to find scallop height h for a given stepover S. From Figure 12.10(a), we have

$$\alpha = \sin^{-1}[(S/2)/R] \tag{12.4a}$$

Thus,

$$h = R\,[1 - \cos \alpha] \tag{12.4b}$$

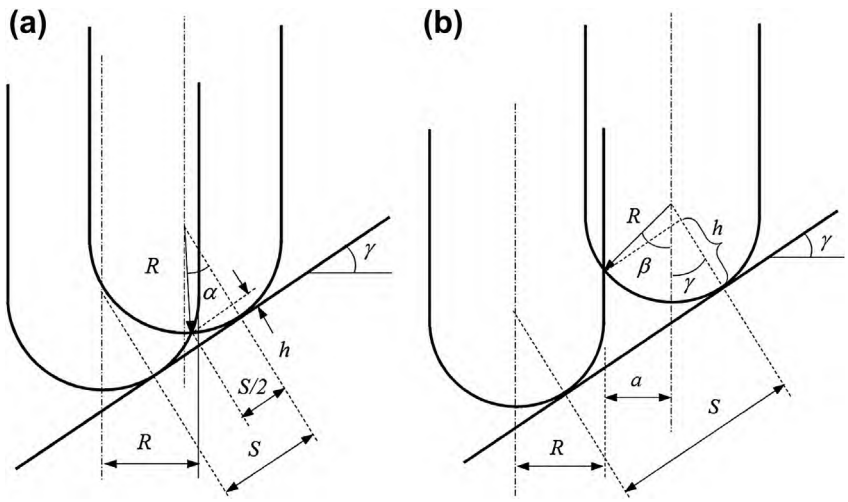


FIGURE 12.10

Illustration of scallop-height calculations: (a) Case 1, $R \cos \gamma > S/2$, and (b) Case 2, $R \cos \gamma < S/2$.

Now, for a given h, we want to find stepover S and number of passes N. Again, referring to Figure 12.10(a), we have

$$\alpha = \cos^{-1}(1 - h/R) \tag{12.5a}$$

Thus,

$$S = 2R\ \sin \alpha \tag{12.5b}$$

And the number of passes N is

$$N = [d/S] + 1 \tag{12.5c}$$

where d is the width of the design surface where the scallop height is calculated, and [•] is an operation that rounds the division to the next greater integer.

CASE 2: $R \cos \gamma < S/2$

Given stepover S, we want to find scallop height h.

From Figure 12.10(b), we have

$$R + a = S \cos \gamma$$

Thus,

$$a = S \cos \gamma - R, \text{ and } \beta = \sin^{-1}(a/R) \tag{12.6a}$$

Therefore, the scallop height is

$$h = R[1 - \cos(\beta + \gamma)] \tag{12.6b}$$

Now, for a scallop height h calculated, we want to find stepover S and number of passes N. Again, referring to Figure 12.10(b), we have,

$$\cos(\beta + \gamma) = (1 - h/R), \text{ and then } \beta = \cos^{-1}(1 - h/R) - \gamma \tag{12.7a}$$

and

$$a = R \sin \beta \tag{12.7b}$$

Thus,

$$S = (R + a)/\cos \gamma \tag{12.7c}$$

and

$$\mathrm{N} = [d/S] + 1 \tag{12.7d}$$

For this example, with the stepover being 1 in. and slope $\gamma = 45^{\circ}$, the condition of Case 2 is satisfied; that is, $R \cos \gamma = 0.6 \cos(45) = 0.4243$ in. $< S/2 = 1/2 = 0.5$ in. Therefore, we use Eqs 12.6a and b to calculate the scallop height; that is,

$$a = S \cos \gamma - R = 1.0 \cos(45) - 0.6 = 0.1072 \text{ in., and}$$
$$\beta = \sin^{-1}(a/R) = \sin^{-1}(0.1072/0.6) = 10.28^{\circ}$$

Therefore, the scallop height is

$$h = R[1 - \cos(\beta + \gamma)] = 0.6[1 - \cos(10.28 + 45)] = 0.2583 \text{ in.}$$

The scallop height is one of the machining parameters that you can define in CAM software (e.g., Pro/MFG) to control the quality of the machined surface. Scallop height determines surface finish by adjusting the number of passes. In this example, how many passes must we have if the scallop height is set to 0.3? The number of passes will still be six since the scallop height of 0.3 in. is larger than the one that resulted in the current number of passes. No change in toolpath will occur.

How about a scallop height of 0.05 in.? How many passes will be generated? Since the required scallop height is much smaller than the current one, the number of passes will increase significantly. We have reason to assume that since the stepover will be small, the case should fall into Case 1. We will use Eq. 12.5 for this calculation and verify this assumption at the end. From Eq. 12.5a, we have

$$\alpha = \cos^{-1}(1 - h/R) = \cos^{-1}(1 - 0.05/0.6) = 23.56^{\circ}$$

thus,

$$S = 2R \sin \alpha = 2(0.6) \sin(23.56) = 0.4796 \text{ in.}$$

And the number of passes N is

$$N = [d/S] + 1 = [5/0.4796] + 1 = 2 + 1 = 12$$

With stepover 0.4796 in., we have $R\cos\gamma = 0.6\cos(45) = 0.4243$ in. $> S/2 = 0.4796/2 = 0.2398$ in.; therefore, Case 1 is verified.

As discussed previously, the tool cuts the surface from one end to the other. There are no intermediate stops in the individual passes. Which are the factors that contribute to the "no intermediate stops" fact? These factors will be discussed in the next subsection.

12.3 RULED SURFACE

In this section, we will move one step further toward studying more complicated and general cases. The design surface for the surface milling sequences in this lesson is a ruled surface.

A ruled surface is defined by two path curves on opposite sides of the surface, the trace of a straight line with its start and end points pass through the respective path curves generating a ruled surface, as illustrated in Chapter 2, Figure 2.23(a). Surfaces, such as flat plane, cone, and cylindrical, can be considered as special cases of ruled surfaces. It is worth noting that a surface with a boundary formed by four straight lines that are not co-planar, as shown in Figure 2.23(b), is not a flat surface but a ruled surface. In this case, both path curves $\boldsymbol{p}$ and $\boldsymbol{q}$ are straight lines, which are not co-planar.

This section focuses on the computational aspect of virtual machining. We will use the block example shown in Figure 12.11 to illustrate the computational methods for both CL data and scallop height. There are three main topics to discuss. First, we will identify factors that affect the toolpath and the quality of the machined surface. These are parameters that you will have to specify carefully in CAM software in order to generate the desired toolpath. Second, we will discuss methods for calculating scallop height on a ruled surface, which determines the quality of the machined surface. Third, we will learn how to generate toolpaths on the ruled surface for both flat-end and ball-nose cutters together with 3-, 4-, and 5-axis mills.

We start with the manufacturing example—a rectangular block with a ruled surface of straight boundary edges. The focus is on calculations of scallop height and number of passes under various conditions and verifying the calculations with those produced by Pro/MFG (if not Pro/MFG, you may use the CAM tool at hand to do the same). We will then calculate toolpaths (i.e., the CL data) and

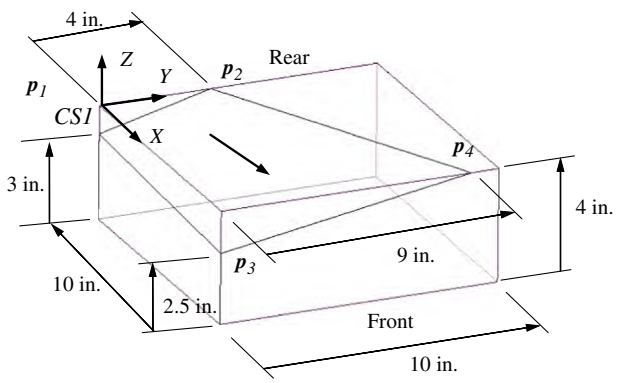


FIGURE 12.11

The assembled design model and workpiece.

verify those again with Pro/MFG. The CL data calculations have been implemented in a spreadsheet, *Chapter 12.3.xls*, which can be found in the companion website of the book.

The block example shown in Figure 12.11 contains both design model and workpiece. The design model is a block with a ruled surface, which is to be machined. The workpiece is a 10 in. × 10 in. × 4 in. block, created in assembly mode with a coordinate system *CS1* (see Figure 12.11), which is chosen as the machine zero.

One sequence (i.e., *contour surface milling*) is defined in this operation. In this sequence, a ball-nose cutter with diameter 1.2 in. and stepover $S = 1$ in. have been employed. Later, we will use a flat-end cutter with the same diameter and length to generate toolpaths. Will the toolpaths generated using these two cutters be different?

12.3.1 5-AXIS MILL WITH BALL-NOSE CUTTER (OP010)

In this subsection, we will discuss the toolpath of the first operation with a ball-nose cutter. Note that the critical machining parameters to be studied include number of passes and scallop height. In addition, we will review the CL data generated by Pro/MFG and determine the impact of the tolerance parameter on the number of CL data along each pass. The toolpath, such as the one shown in Figure 12.12(a), can be generated by Pro/MFG. There are eight passes. Note that the stepover defined is 1 in. Is the toolpath satisfying our requirement in terms of the stepover?

The lengths of the edges p_1p_2 and p_3p_4 are $d_{12} = \sqrt{4^2 + 1^2} = 4.123$ in. and $d_{34} = \sqrt{9^2 + 1.5^2} = 9.124$ in., respectively. The actual distances between passes at edges p_1p_2 and p_3p_4 are 0.590 in. ($d_{12}/7$) and 1.303 in. ($d_{34}/7$), respectively. Note that the number of intervals is 7; that is, the number of passes is 8 minus 1. The distance between passes at edges p_1p_2 is less than the specified stepover; however, that is not the case for edge p_3p_4.

12.3.1.1 Number of Passes

The milling simulation illustrated in Figure 12.12(b) shows that the passes are too far apart at edge p_3p_4; thus, material is left uncut. It is obvious that Pro/MFG does not properly determine the number of passes at this edge.

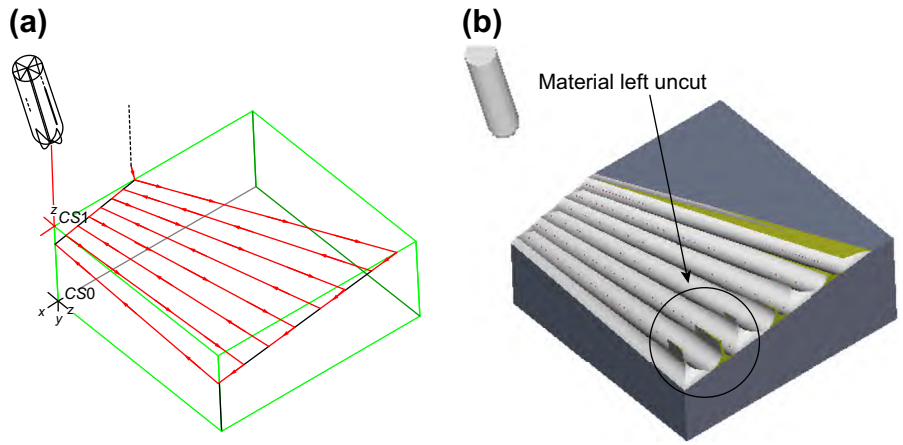


FIGURE 12.12

OP010 operation: (a) toolpath and (b) machining simulation with material left on the design surface.

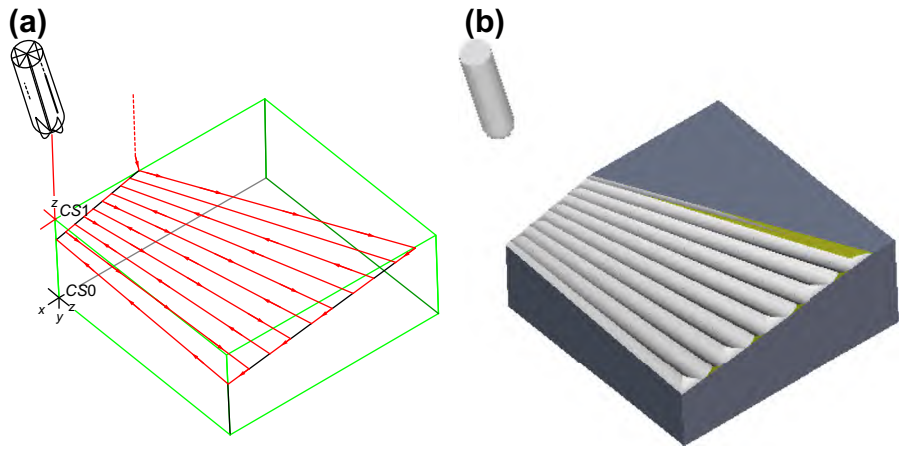


FIGURE 12.13

OP010 operation with ten passes: (a) toolpath and (b) machining simulation.

In fact, the stepover specified in Pro/MFG is imposed on the mid-plane of the design surface. The length of the straight line at the mid-plane can be calculated by first calculating the width and height; that is, width = (1 + 1.5)/2 = 1.25 and height = (4 + 9)/2 = 6.5. As a result, the length is $d_m = \sqrt{1.25^2 + 6.5^2} = 6.619$ in. Note that, by definition, the straight line will lie on the ruled surface. The actual distance between passes across this straight line is 0.946 in. (i.e., 6.619/7), which meets the required stepover. Therefore, *for a ruled surface of uneven edge lengths, Pro/MFG generates number of passes that satisfies the prescribed stepover at the mid-plane.* This may or may not be true for the CAM software you are using. However, this study should provide you with a good understanding in determining the number of passes in the toolpath.

Note that you may also specify the number of passes in Pro/MFG (and other CAM software). Will the toolpath be different if you specify the number of passes as 5? The answer is "no" since the minimum number of passes required to satisfy the prescribed stepover is 8, as discussed earlier. Anything less than 8 will not make any difference in toolpath.

If you specify the number of passes to be 10, Pro/MFG will give you 10 passes, as shown in Figures 12.13(a) and (b). Note that the scallop at the front end (i.e., edge p_3p_4) is significantly reduced.

12.3.1.2 Scallop Height

As shown in Figure 12.14, the distance h specifies the scallop height of the surface. What is the maximum scallop height in the NC sequence shown in Figure 12.13? Apparently, the scallop height varies on the design surface. The scallop height at the front edge (p_3p_4) is larger than that at the rear edge (p_1p_2). How is it possible to calculate the scallop height? How does CAM software (e.g. Pro/MFG) take the prescribed scallop height to determine the number of passes?

For this NC sequence (i.e., 5-axis mill with a ball-nose cutter) the scallop height can be calculated as follows, assuming that the number of passes is 10. Note that in a 5-axis mill with a ball-nose cutter, the cutter is normal to the design surface. This is assumed in Pro/MFG. However, for other CAM tools,

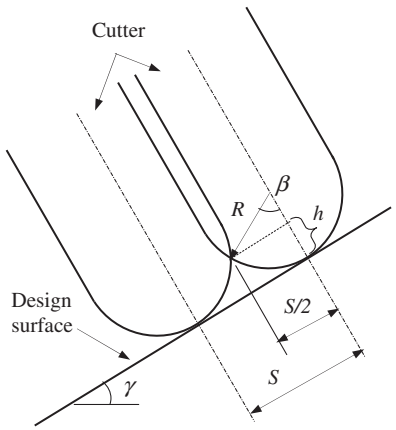


FIGURE 12.14

Illustration of scallop-height calculation for *OP010*.

such as Mastercam, users may add a small angle to tilt the cutter from the direction normal to the design surface.

As illustrated in Figure 12.14, the scallop height h is

$$h = R(1 - \cos\beta) \tag{12.8a}$$

where

$$\beta = \sin^{-1}[S/(2R)] \tag{12.8b}$$

Note that R is the radius of the cutter, and S is the stepover. At the front edge, p_3p_4, the stepover is

$$S = d_{34}/(\text{number of passes} - 1) = 9.124/9 = 1.014''$$

and

$$\beta = \sin^{-1}[S/(2R)] = \sin^{-1}[1.014/(2 \times 0.6)] = 57.7^\circ$$

Thus, the scallop height is

$$h = R\big(1 - \cos\beta\big) = 0.6\big(1 - \cos 57.7^\circ\big) = 0.279''$$

Similarly, on the rear edge p_1p_2, the scallop height is 0.0454 in. Also, at the mid-plane, the edge length is 6.619 in., as discussed before. The scallop height at the mid-plane is therefore 0.126 in.

You may also specify a scallop height in Pro/MFG to control the quality of the machined surface. What is the scallop height that will be just large enough to cause increments in the number of passes?

As observed earlier, Pro/MFG generates the number of passes that satisfies the prescribed stepover at the mid-plane. It is also true that *Pro/MFG enforces the prescribed scallop height at the mid-plane*.

Therefore, if you specify a scallop greater than 0.126 in. (e.g., 0.13 in.) nothing will change since the current 10-pass toolpath yields a scallop height less than 0.13 in. However, a prescribed scallop

height less than 0.126 in. will yield more passes. For example, a 0.12 in. scallop height, which is slightly smaller than 0.126 in., will generate 11 passes. How many passes will be generated, if you specify a scallop height 0.02 in.?

Referring to Figure 12.14, given scallop height h, the stepover S can be calculated as,

$$S = 2R \ \sin\beta \tag{12.9a}$$

where

$$\beta = \cos^{-1}(1 - h/R) \tag{12.9b}$$

Thus, the number of passes N is

$$N = [d/S] + 1 \tag{12.9c}$$

Therefore, considering the mid-plane, we have

$$\beta = \cos^{-1}(1 - h/R) = \cos^{-1}(1 - 0.02/0.6) = 14.8^\circ$$

and

$$S = 2R \ \sin\beta = 2 \times 0.6 \times \sin 14.8^\circ = 0.307''$$

Thus,

$$N = [d_m/S] + 1 = [6.619/0.307] + 1 = 22 + 1 = 23$$

Give it a try in Pro/MFG. You should see the toolpath like that of Figure 12.15.

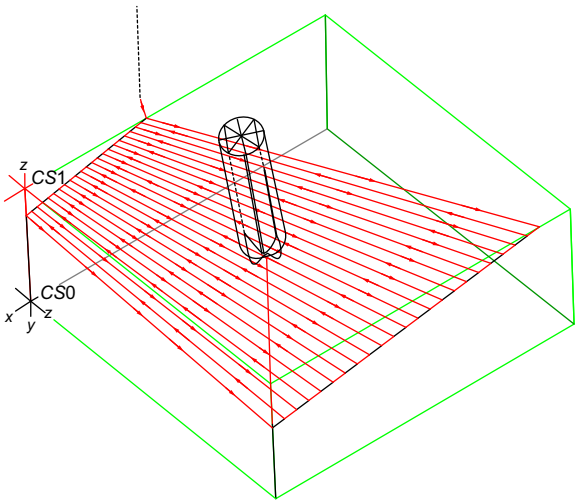


FIGURE 12.15

Toolpath for scallop height $h = 0.02$ in. This figure is reproduced in color on the book's companion website: http://booksite.elsevier.com/9780123820389.

12.3.1.3 Parametric Surface and CL Data

In the previous section, we discussed CC (cutter contact) and CL (cutter location). In this example, do CC and CL lines coincide? The answer is "yes" since we assume a 5-axis mill, in which cutter orients in a direction normal to the design surface (without additional tilting angle) and a ball-nose cutter. Moreover, the design surface has a small curvature that allows the cutter to adjust its orientation freely so that the contact point and the center of the cutter (CL point) always coincide.

In addition, if you pay attention to the number of stops (CL data) along each pass, you should realize that the cutter does not go all the way from one end to the other. It makes several stops in one pass (along one CC line). Why cannot the cutter go from one end to the other without intermediate stops? How does one calculate the CL data? What factors affect CL data?

Note that the CL data include location of the tip of the cutter and orientation of the cutter. If you open a CL data file generated by Pro/MFG (or a CAM tool you use), you should see that each CL data point consists of *X*-, *Y*-, and *Z*-locations, as well as *X*-, *Y*-, and *Z*-components of the normalized vector that specifies the orientation of the cutter. For this operation, the cutter orientation vector is normal to the design surface in Pro/MFG.

Before answering these questions and calculating CL data, we must represent the design surface mathematically, which is a ruled surface in this example. As discussed in Chapter 2, the *X*-, *Y*-, and *Z*-coordinates of a given point on the parametric surface can be represented as functions of two parameters u and w, as shown in Figure 12.16(a). Note that usually $u \in [0,1]$ and $w \in [0,1]$.

The sweeping straight line along the surface actually connects points on the two path curves with the same u-parametric value. For example, the straight line that connects the mid-points of $\boldsymbol{p}(u)$ and $\boldsymbol{q}(u)$ at $u = 0.5$ stays right on the ruled surface.

The Cartesian coordinates of any given point on the surface can be obtained by specifying its corresponding parametric coordinates (i.e., u and w values) in the mathematical equations of the parametric surface, which will be discussed next. Note that the parametric representation of geometric entities is very powerful and is widely used for geometric modeling and solid modeling.

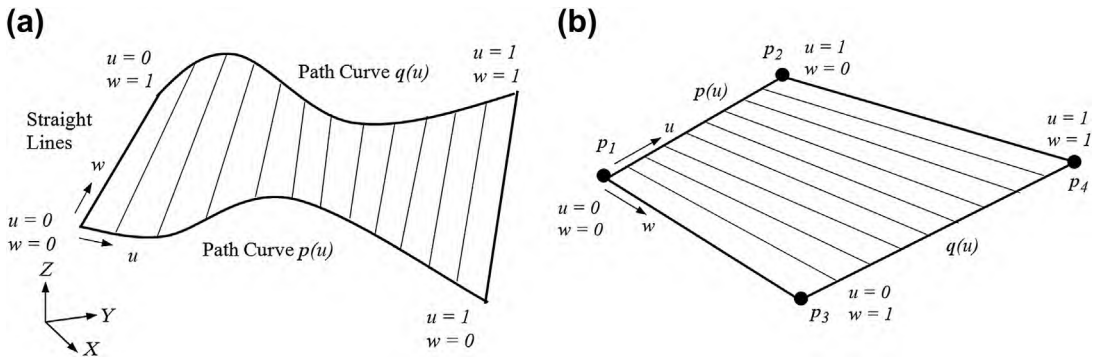


FIGURE 12.16

Parametric representation of a ruled surface: (a) general ruled surface and (b) block example.

Since both path curves are straight lines in this example, we start with the mathematic representation of parametric straight lines. More in-depth discussion regarding parametric curves and surfaces can be found in Chapter 2.

Given two distinct points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ in space, the straight line that connects these two points can be written as

$$\boldsymbol{p}(u) = \big[p_x(u), p_y(u), p_z(u)\big] = (1-u)\boldsymbol{p}_1 + u\,\boldsymbol{p}_2 = (1-u)\big[p_{1x}, p_{1y}, p_{1z}\big] + u\big[p_{2x}, p_{2y}, p_{2z}\big] \quad (12.10)$$

where $u \in [0,1]$. Hence, the parametric equations of the path curves $\boldsymbol{p}(u)$ and $\boldsymbol{q}(u)$ of the design surface of the block example, as shown in Figure 12.16(b), are

$$\boldsymbol{p}(u) = (1-u)\boldsymbol{p}_1 + u\,\boldsymbol{p}_2$$
$$\boldsymbol{q}(u) = (1-u)\boldsymbol{p}_3 + u\,\boldsymbol{p}_4$$

where

$$\boldsymbol{p}_1 = [0, 0, -1]$$
$$\boldsymbol{p}_2 = [0, 4, 0]$$
$$\boldsymbol{p}_3 = [10, 0, -1.5]$$
$$\boldsymbol{p}_4 = [10, 9, 0]$$

following the machine zero *CS*1 defined on the top surface of the workpiece (see Figure 12.11).

Next, we will formulate the parametric equations to represent the ruled surface shown in Figure 12.16(b). The ruled surface $\boldsymbol{s}(u,w)$ is

$$\begin{aligned}
\boldsymbol{s}(u,w) &= [s_x(u,w), s_y(u,w), s_z(u,w)] \\
&= \boldsymbol{p}(u) + w[\boldsymbol{q}(u) - \boldsymbol{p}(u)] = (1-w)\boldsymbol{p}(u) + w\boldsymbol{q}(u) \\
&= (1-w)[(1-u)\boldsymbol{p}_1 + u\,\boldsymbol{p}_2] + w[(1-u)\boldsymbol{p}_3 + u\,p_4] \\
&= (1-w)\{(1-u)[0,0,-1] + u[0,4,0]\} + w\{(1-u)[10,0,-1.5] + u[10,9,0]\} \\
&= [10w, 4u + 5uw, -1 + u - 0.5w + 0.5uw]
\end{aligned} \quad (12.11)$$

Thus, for any given point on the surface with $u \in [0,1]$ and $w \in [0,1]$, its Cartesian coordinates can be calculated by using Eq. 12.11. Note that when we fixed a w value, the surface equations degenerate into a curve (or a straight line in our case); that is,

$$\boldsymbol{s}(u, w_i) = (1-w_i)\boldsymbol{p}(u) + w_i\,\boldsymbol{q}(u) \quad (12.12)$$

For example, if $w = 0$, $\boldsymbol{s}(u,0) = \boldsymbol{p}(u)$; and when $w = 1$, $\boldsymbol{s}(u,1) = \boldsymbol{q}(u)$. Similarly, when we fix a u value, the surface equations degenerate into a straight line that connects points on $\boldsymbol{p}(u_i)$ and $\boldsymbol{q}(u_i)$, respectively; that is,

$$\begin{aligned}
\boldsymbol{s}(u_i, w) &= (1-w)\boldsymbol{p}(u_i) + w\,\boldsymbol{q}(u_i) \\
&= (1-w)\big[(1-u_i)\boldsymbol{p}_1 + u_i\,\boldsymbol{p}_2\big] + w\big[(1-u_i)\boldsymbol{p}_3 + u_i\,\boldsymbol{p}_4\big]
\end{aligned} \quad (12.13)$$

For example, when $u = 0$,

$$\boldsymbol{s}(0, w) = (1 - w)\big[(1 - 0)\,\boldsymbol{p}_1 + 0\,\boldsymbol{p}_2\big] + w\big[(1 - 0)\,\boldsymbol{p}_3 + 0\,\boldsymbol{p}_4\big] = (1 - w)\,\boldsymbol{p}_1 + w\,\boldsymbol{p}_3$$

which is a straight line that connects points $\boldsymbol{p}_1$ and $\boldsymbol{p}_3$. Note that, as shown in Figure 12.13(a), the CC lines in this example are evenly distributed along the u-direction.

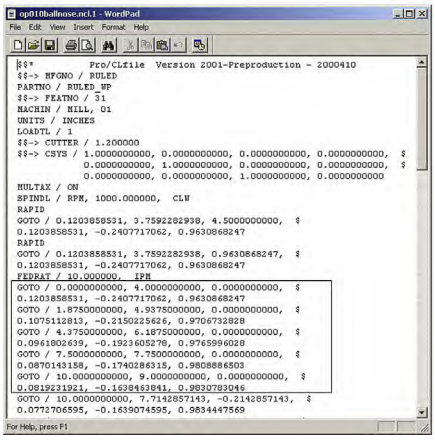
As a result, the CC lines are simply the straight lines at $u = 0$, 1/9, 2/9, 1/3, 4/9, 5/9, 2/3, 7/9, 8/9, and 1, respectively, since the number of passes is 10. Note that in this example, the first pass goes from $\boldsymbol{p}_2$ to $\boldsymbol{p}_4$ (i.e., at $u = 1$). Therefore, any CL data along the straight line (the first pass at $u = 1$) can be obtained by locating a w-value on the surface equations; that is,

$$\begin{aligned}\boldsymbol{s}(1, w_i) &= (1 - w_i)\,\boldsymbol{p}(1) + w_i\,\boldsymbol{q}(1)\\ &= (1 - w_i)\big[(1 - 1)\,\boldsymbol{p}_1 + 1\,\boldsymbol{p}_2\big] + w_i\big[(1 - 1)\,\boldsymbol{p}_3 + 1\,\boldsymbol{p}_4\big] = (1 - w_i)\,\boldsymbol{p}_2 + w_i\,\boldsymbol{p}_4 \qquad (12.14)\\ &= (1 - w_i)[0, 4, 0] + w_i[10, 9, 0] = [10w_i, 4 + 5w_i, 0]\end{aligned}$$

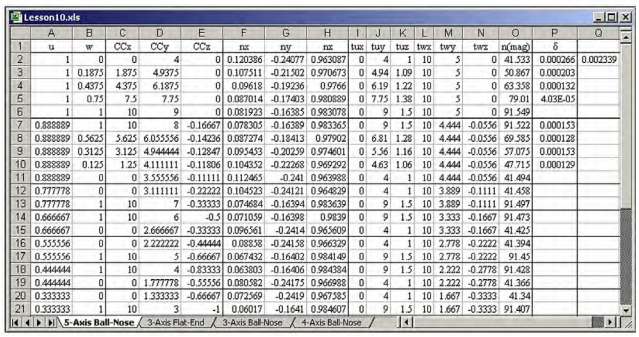
The X-, Y-, and Z-coordinates of the first five CL data generated by Pro/MFG are shown in Figure 12.17(a). The CL data can be found in the file *op010.ncl* at the book's companion site. Note that the first three data after the *GOTO* command are X-, Y- and Z-coordinates of the CL point. The next three specify the orientation of the cutter, which will be discussed later.

The same location data X, Y, and Z can be obtained by plugging $w = 0$, 0.1875, 0.4375, 0.75, and 1, respectively, into Eq. 12.14. The CL data can be easily calculated using a spreadsheet, as implemented in the *Chapter 12.4.xls* file at the book's companion site. A portion of the data are listed in Columns C

(a)



(b)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	u	w	CCx	CCy	CCz	nx	ny	nz	tux	tuy	tuz	twx	twy	twz	n(mag)	δ	
2	1	0	0	4	0	0.120386	-0.24077	0.963087	0	4	1	10	5	0	41.533	0.000266	0.002339
3	1	0.1875	1.875	4.9375	0	0.107511	-0.21502	0.970673	0	4.94	1.09	10	5	0	50.867	0.000203	
4	1	0.4375	4.375	6.1875	0	0.09618	-0.19236	0.9766	0	6.19	1.22	10	5	0	63.358	0.000132	
5	1	0.75	7.5	7.75	0	0.087014	-0.17403	0.980839	0	7.75	1.38	10	5	0	79.01	4.03E-05	
6	1	1	10	9	0	0.081923	-0.16385	0.983078	0	9	1.5	10	5	0	91.549		
7	0.888889	1	10	8	-0.16667	0.078305	-0.16389	0.983365	0	9	1.5	10	4.444	-0.0556	91.522	0.000153	
8	0.888889	0.5625	5.625	6.055556	-0.14236	0.087274	-0.18413	0.97902	0	6.81	1.28	10	4.444	-0.0556	69.585	0.000128	
9	0.888889	0.3125	3.125	4.944444	-0.12847	0.095453	-0.20239	0.974601	0	5.56	1.16	10	4.444	-0.0556	57.075	0.000153	
10	0.888889	0.125	1.25	4.111111	-0.11806	0.104352	-0.22268	0.969292	0	4.63	1.06	10	4.444	-0.0556	47.715	0.000129	
11	0.888889	0	0	3.555556	-0.11111	0.112465	-0.241	0.963988	0	4	1	10	4.444	-0.0556	41.494		
12	0.777778	0	0	3.111111	-0.22222	0.104523	-0.24121	0.964829	0	4	1	10	3.889	-0.1111	41.458		
13	0.777778	1	10	7	-0.33333	0.074684	-0.16394	0.983639	0	9	1.5	10	3.889	-0.1111	91.497		
14	0.666667	1	10	6	-0.5	0.071059	-0.16398	0.9839	0	9	1.5	10	3.333	-0.1667	91.473		
15	0.666667	0	0	2.666667	-0.33333	0.096561	-0.2414	0.965609	0	4	1	10	3.333	-0.1667	41.425		
16	0.555556	0	0	2.222222	-0.44444	0.08838	-0.24138	0.966329	0	4	1	10	2.778	-0.2222	41.394		
17	0.555556	1	10	5	-0.66667	0.067432	-0.16402	0.984149	0	9	1.5	10	2.778	-0.2222	91.45		
18	0.444444	1	10	4	-0.83333	0.063803	-0.16406	0.984384	0	9	1.5	10	2.222	-0.2778	91.428		
19	0.444444	0	0	1.777778	-0.55556	0.080582	-0.24175	0.966988	0	4	1	10	2.222	-0.2778	41.366		
20	0.333333	0	0	1.333333	-0.66667	0.072569	-0.2419	0.967585	0	4	1	10	1.667	-0.3333	41.34		
21	0.333333	1	10	3	-1	0.06017	-0.1641	0.984607	0	9	1.5	10	1.667	-0.3333	91.407		

FIGURE 12.17

Toolpath verification for *OP010*: (a) CL data file generated by Pro/MFG, and (b) spreadsheet calculation.

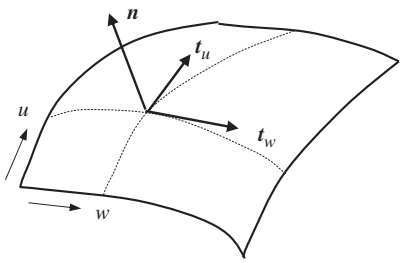


FIGURE 12.18

A normal vector of a general parametric surface.

to E, as shown in Figure 12.17(b). Compare the data points in the spreadsheet with those generated by Pro/MFG. They are identical!

Note that since we employed a 5-axis mill as the workcell, CL data also contain cutter orientation data. A normalized vector that is normal to the design surface defines the cutter orientation. We must calculate the normal vector of the ruled surface.

The normal vector of a given parametric surface in space, such as the one shown in Figure 12.18, can be calculated by,

$$\boldsymbol{n} = \frac{\boldsymbol{t}_w \times \boldsymbol{t}_u}{|\boldsymbol{t}_w \times \boldsymbol{t}_u|} \tag{12.15}$$

where $\boldsymbol{t}_w$ and $\boldsymbol{t}_u$ are tangent vectors of a given point on the surface along the w and u directions, respectively. The tangent vectors are defined as

$$\boldsymbol{t}_w = \boldsymbol{s}(u,w)_{,w} = \frac{\partial \boldsymbol{s}(u,w)}{\partial w} \tag{12.16a}$$

and

$$\boldsymbol{t}_u = \boldsymbol{s}(u,w)_{,u} = \frac{\partial \boldsymbol{s}(u,w)}{\partial u} \tag{12.16b}$$

For this example, we have

$$\boldsymbol{t}_w = \boldsymbol{s}(u,w)_{,w} = \left[10, 5u, -0.5 + 0.5u\right]$$

and

$$\boldsymbol{t}_u = \boldsymbol{s}(u,w)_{,u} = \left[0, 4 + 5w, 1 + 0.5w\right]$$

Thus,

$$\boldsymbol{n} = \frac{\boldsymbol{t}_w \times \boldsymbol{t}_u}{|\boldsymbol{t}_w \times \boldsymbol{t}_u|} = \frac{[2 + 3u + 2.5w, -10 - 5w, 40 + 50w]}{\sqrt{(2 + 3u + 2.5w)^2 + (-10 - 5w)^2 + (40 + 50w)^2}} \tag{12.17}$$

Since

$$\boldsymbol{t}_w \times \boldsymbol{t}_u = \begin{vmatrix} i & j & k \\ 10 & 5u & -0.5 + 0.5u \\ 0 & 4 + 5w & 1 + 0.5w \end{vmatrix}$$

$$= [5u(1 + 0.5w) - (-0.5 + 0.5u)(4 + 5w)]i + [0 - 10(1 + 0.5w)]j + [10(4 + 5w) - 0]k$$

$$= [2 + 3u + 2.5w, -10 - 5w, 40 + 50w]$$

Again, the cutter orientation vector can be calculated using a spreadsheet. A portion of the data are listed in Columns F to H, as shown in Figure 12.17(b). Compare the data points in the spreadsheet with the CL data generated by Pro/MFG. Again, they are identical!

12.3.1.4 A Few Questions

Next, we will discuss a few critical questions regarding the CL data. First, why are there more than one CL points along each pass; that is, a straight line that is defined by $\boldsymbol{s}(u_i,w)$? Are these CL points collinear? Do they lie on the straight line defined by $\boldsymbol{s}(u_i,w)$?

Yes, the CL data points along a tool pass are collinear. They are located on the same straight line (see rows 2 to 6 of Figure 12.17(b)). How can you tell? (They all share the same slope of the CC line they reside.)

Why are there CL data points between end points along a pass? Let us look at the equation of the normal vector $\boldsymbol{n}$, defined in Eq. 12.17. For the first pass (i.e., when $u = 1$), we have

$$\boldsymbol{n} = \frac{\boldsymbol{t}_w \times \boldsymbol{t}_u}{|\boldsymbol{t}_w \times \boldsymbol{t}_u|} = \frac{[5 + 2.5w, -10 - 5w, 40 + 50w]}{\sqrt{(5 + 2.5w)^2 + (-10 - 5w)^2 + (40 + 50w)^2}}$$

As shown in the equation above the normal vector $\boldsymbol{n}$ is not constant along the first pass (and along any other CC lines in this example). It is a function of w. Is it a linear function of w? No, it is a nonlinear rational function. Pro/MFG (and other CAM tools) must discretize the tool pass to approximate the varying normal vector. Why? It is because the controller on a CNC mill cannot orient the cutter following continuous functions determined by the true normal vector. It must set a cutter orientation, moving from one CL to the next CL with the same orientation and then reorient the cutter before moving to the next stop.

Now, the second question is, can Pro/MFG afford not to adjust the cutter orientation along a pass? The answer is no. Overcut or undercut will occur if the cutter is not reoriented. For example, let's look at the first pass $\boldsymbol{p}_2$ to $\boldsymbol{p}_4$ (see Figure 12.19(a)), where $u = 1$. At point A, the cutter is oriented according to the surface normal at A, as illustrated in Figure 12.19(b).

Note that the surface slopes at points A and B are exaggerated for illustration purpose. If the cutter is not reoriented, it will cause excessive cut (or interference) at point B, as illustrated in Figure 12.19(b). Note that an interference of 0.002339 in. exists if the cutter orientation is not adjusted while moving from $\boldsymbol{p}_2$ to $\boldsymbol{p}_4$ in the block example. For details, see *Cell Q2* in the *5-Axis Ball-Nose* worksheet of the spreadsheet on the companion site (see file *Chapter 12.4.xls*).

How to calculate the interference? As illustrated in Figure 12.19(c), two vectors, $\boldsymbol{R}_A$ and $\boldsymbol{R}_B$, are defined, where $\boldsymbol{R}_A$ is the vector from the center of the cutter to the tool tip (i.e., point B). This vector

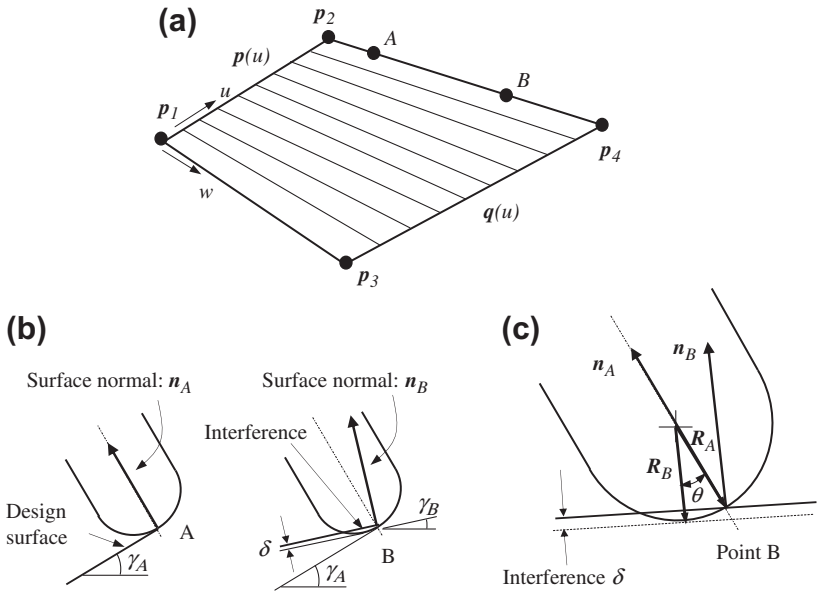


FIGURE 12.19

Adjustment of cutter orientation to avoid interference: (a) intermediate points A and B, (b) cutter orientation at points A and B, and (c) illustration for interference computation.

is parallel to $\boldsymbol{n}_A$ but points to the opposite direction. $\boldsymbol{R}_B$ is the vector from the center of the cutter to the tool tip, which is also the deepest point cut into the workpiece. This vector is parallel to $\boldsymbol{n}_B$ but points in the opposite direction. Note that $\boldsymbol{R}_A = -R\,\boldsymbol{n}_A$ and $\boldsymbol{R}_B = -R\,\boldsymbol{n}_B$. Therefore, the angle between these two vectors is

$$\cos\theta = \frac{\boldsymbol{R}_A \cdot \boldsymbol{R}_B}{|\boldsymbol{R}_A||\boldsymbol{R}_B|} = \frac{\boldsymbol{R}_A \cdot \boldsymbol{R}_B}{R^2} = \boldsymbol{n}_A \cdot \boldsymbol{n}_B$$

Hence the interference δ can be calculated by

$$\delta = R(1 - \cos\theta) = R(1 - \boldsymbol{n}_A \cdot \boldsymbol{n}_B) \tag{12.18}$$

How do we impose the requirement on the interference to minimize the excessive cut? How is it possible to minimize the error due to the excessive cut? The amount of allowable excessive cut can be specified in the parameter *TOLERANCE*.

In Pro/MFG, the default value of the tolerance parameter is 0.001. What will happen if you reduce the tolerance from 0.001 to 0.0001? More stops will be inserted into each pass to reorient the cutter.

Check *Column P* of the *5-Axis Ball-Nose* worksheet of the spreadsheet *Chapter 12.4.xls* for sample tolerance data. Note that the tolerance data computed are much smaller than the *TOLERANCE* parameter required (i.e., 0.001).

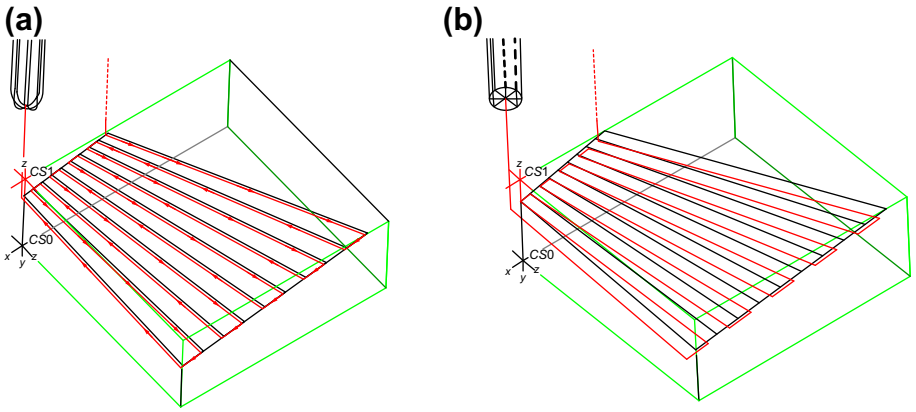


FIGURE 12.20

A 3-axis mill toolpath: (a) ball-nose cutter and (b) flat-end cutter.

12.3.2 3-AXIS MILL WITH FLAT-END CUTTER (OP030)

In this subsection, we will employ a 3-axis mill to conduct a surface contour milling for the same ruled surface. If you choose the ball-nose cutter as defined in *OP030* and *NC Sequence: 1: Contour Surface Milling*, you should see that the toolpath generated by Pro/MFG looks like the one shown in Figure 12.20(a). Is the toolpath different from that of the 5-axis mill defined in operation *OP010*? Is there a need to adjust the cutter orientation for the 3-axis mill? No, since the cutter orientation is fixed—always aligns with the *Z*-axis of the machining coordinate system—for a 3-axis mill.

By observing the toolpath shown in Figure 12.20(a), we ask the following questions:

1. Why does Pro/MFG generate the toolpath like the one shown in Figure 12.20(a), where CC and CL curves do not coincide?
2. There are five CL data points on the first pass ($u = 1$). Are these five points collinear?
3. How are these five points determined? Which factors will affect the number of CL data points along a pass?
4. Why can't the cutter just move from one end to the other with intermediate stops?

Before we answer these questions, let's change the tool to a flat-end cutter. This cutter yields a toolpath that is easier to understand. We will come back to the ball-nose cutter and revisit the preceding questions later.

12.3.2.1 Flat-End Cutter

The diameter and length of the flat-end cutter are 1.2 in. and 2 in., respectively. The toolpath generated using the flat-end cutter is illustrated in Figure 12.20(b). We can see the toolpath more clearly from the top view (Figure 12.21).

As shown in Figures 12.20(b) and 12.21(a), the CC and CL lines do not coincide. Are there any intermediate points along a pass? No, the cutter moves from one end to the other without intermediate

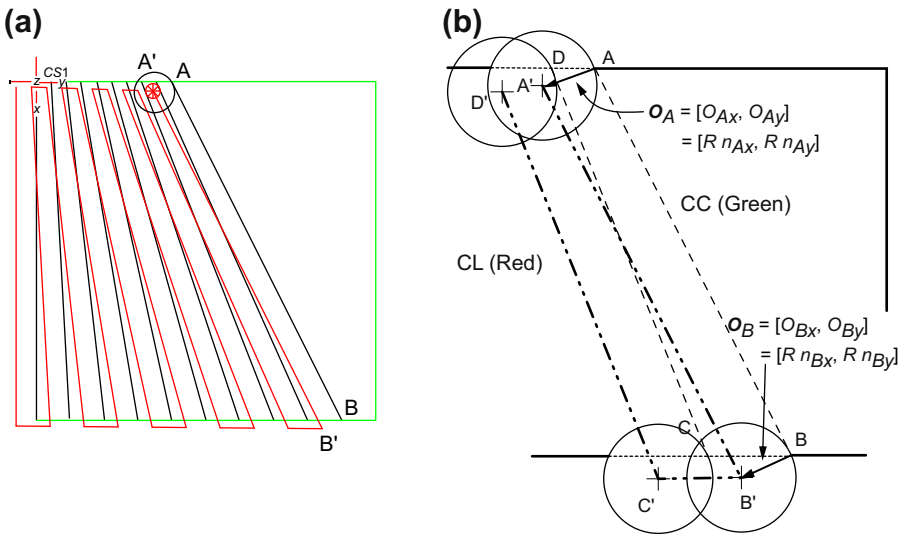


FIGURE 12.21

A 3-axis toolpath from top view: (a) toolpath in Pro/MFG and (b) illustration for offset calculations. This figure is reproduced in color on the book's companion website: http://booksite.elsevier.com/9780123820389.

stops, which is different from what we saw in the previous case. Why is there no intermediate stop? Will there be excessive cuts? We will discuss these questions next.

12.3.2.1.1 Flat-End Cutter Toolpath Generation

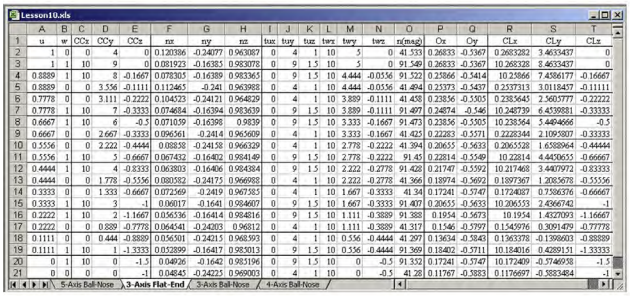
The first observation of the toolpath shown in Figure 12.21(a) is that the outer edge of the cutter is always tangent to the CC line. This is one of the fundamental principles that the parametric method (implemented in Pro/MFG) follows while generating a toolpath. This principle was also observed in operation *OP010*, where the center point of the cutter tip is always in contact and is tangent to the CC lines. The implication of this principle is that the cutter always cuts the surface "just right", not less, not more, yielding the best possible cut.

Second, the stepover (e.g., B′C′ in Figure 12.21(b)) is identical to the distance between two CC curves (e.g., BC in Figure 12.21(b)) all the time. This must be the case if the first observation stands.

Third, the CC and CL curves do not coincide. This again results from the first observation. The CL data point is offset from the CC point horizontally; that is, it is at the same elevation along the *Z*-direction of the machining coordinate system *CS1*. Thus, the *Z*-offset is zero. The magnitude of the horizontal offset is identical to the radius of the cutter in order to maintain tangency at the cutter contact. The direction of the horizontal offset O_{xy} is normal to the CC line on the *X*-*Y* plane. Therefore, the offset can be calculated by

$$O_x = \frac{R\, n_x}{\sqrt{n_x^2 + n_y^2}} \tag{12.19a}$$

(a)



(b)

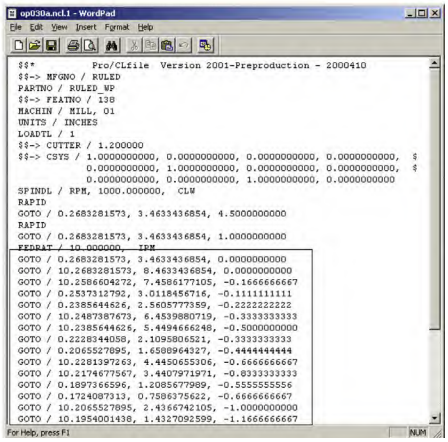


FIGURE 12.22

Toolpath verification for *OP030*: flat-end cutter: (a) spreadsheet calculation and (b) CL data file generated by Pro/MFG.

$$O_y = \frac{R\, n_y}{\sqrt{n_x^2 + n_y^2}} \tag{12.19b}$$

where n_x and n_y are the components of the normalized vector $\boldsymbol{n}$ that is normal to the surface at the CC point projected on the X-Y plane. A more detailed illustration is given in Figure 12.21(b), in which two passes AB and CD are shown. The corresponding CL data points at A, B, C, and D are labeled A′, B′, C′, and D′, respectively. Note that all the CL data points are offset with an amount of cutter radius on the horizontal plane along a direction that is normal to the CC line projected on the X-Y plane. The components of the offset are not identical at different passes since the normal vector varies.

The CL data can be obtained by the following,

$$\begin{aligned} \mathrm{CL}_x &= \mathrm{CC}_x + \mathrm{O}_x \\ \mathrm{CL}_y &= \mathrm{CC}_y + \mathrm{O}_y \\ \mathrm{CL}_z &= \mathrm{CC}_z \end{aligned} \tag{12.20}$$

Note that Columns R, S, and T of the second worksheet, *3-Axis Flat-End*, of *Chapter 12.4.xls* show the CL data points (Figure 12.22(a)). The offsets O_x and O_y are calculated and listed in Columns P and Q, respectively. Compare the CL data in the spreadsheet with the CL data generated by Pro/MFG (see Figure 12.22(b)). They are identical!

Note that no intermediate stop is needed. Why? It is because that for 3-axis milling with a flat-end cutter, the tangent contact point is always at the cutter corner, and there is nothing to adjust in cutter orientation. The scallop-height calculation is identical to that of Section 12.2, using Eq. 12.1.

12.3.3 3-AXIS MILL WITH BALL-NOSE CUTTER (OP030)

Again, the key principle in generating a toolpath (CL data) is that the CC points always must be tangent to the cutter profile. Based on this principle, we will discuss the calculation of the CL data for this NC sequence.

12.3.3.1 CL Data

The toolpath of this NC sequence generated by Pro/MFG is shown in Figure 12.23. For any given point on the green line, we can calculate the CL by the following.

The key is to determine the tangent contact point on the cutter. Let us cut a view plane that is normal to a given CC line, as shown in Figure 12.24(a). If we view the workpiece through a direction that is normal to the cut plane, as illustrated in Figure 12.24(b), we observe that the CL point is offset from the CC point by

$$O_{xy} = R\left(\frac{\sqrt{n_x^2 + n_y^2}}{\sqrt{n_x^2 + n_y^2 + n_z^2}}\right) = R\sqrt{n_x^2 + n_y^2} \tag{12.21}$$

and

$$\begin{aligned} O_x &= R\, n_x \\ O_y &= R\, n_y \\ O_z &= -R\,(1 - n_z) \end{aligned} \tag{12.22}$$

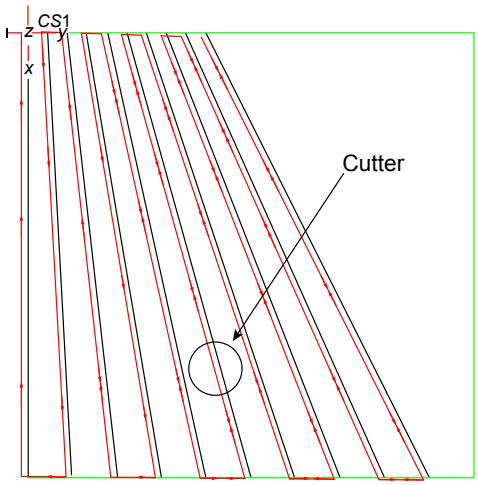


FIGURE 12.23

Toolpath of OP030: ball-nose cutter (top view). This figure is reproduced in color on the book's companion website: http://booksite.elsevier.com/9780123820389.

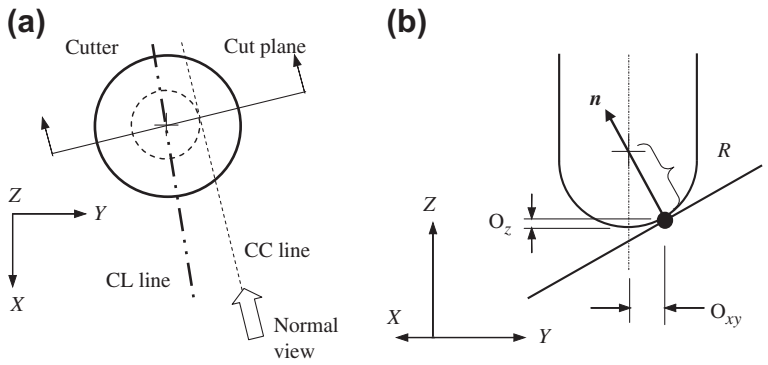


FIGURE 12.24

Illustration for offset calculations: (a) top view with the cut plane and (b) normal view from the cut plane.

where O_{xy} is the horizontal offset of the cutter on the X-Y plane, and O_z is the vertical offset along the Z-direction. Note that n_x, n_y, and n_z are components of the normal vector $\boldsymbol{n}$ in the X-, Y-, and Z-directions, respectively.

Therefore, the CL data can be obtained as

$$\begin{aligned} CL_x &= CC_x + O_x \\ CL_y &= CC_y + O_y \\ CL_z &= CC_z + O_z \end{aligned} \tag{12.23}$$

Part of the CL data is calculated and shown in Figure 12.25(a) (Columns S, T, and U). These results are identical to those of the CL data generated by Pro/MFG, as shown in Figure 12.25(b).

Let us get back to the questions we asked in the beginning of Section 12.3.1. Let us start with Question 4: Why cannot the cutter just move from one end to the other with a proper offset from the CC line?

The answer to this question can be illustrated using the example shown in Figure 12.26. The CL points must be adjusted to avoid excessive cut (remember the orientation of the cutter is fixed in this case). Again, the key is that the cutter always must be tangent to the CC line.

Now, let us address Question 3: Which factors will affect the number of CL points along a pass?

With a 3-axis mill and the ball-nose cutter chosen in this case, two critical factors that affect the number of CL data are the slope of the design surface and the allowable tolerance. The slope of the design surface varies along a given pass.

If the slope is a constant along a pass, as shown in the examples in Section 12.2, there is no need to adjust the cutter offset; therefore, no intermediate CL data points are needed. As shown in Figure 12.26, excessive cuts occur along the pass from point to point. When we reduce the allowable tolerance, more CL data points will be added to the pass, therefore, generating a more accurate machined surface.

Note that the corner radius r of the cutter also will affect the cutter offsets, thus, the CL data. As illustrated in Figure 12.27, the cutter contact point will always lie on the round contour corner of

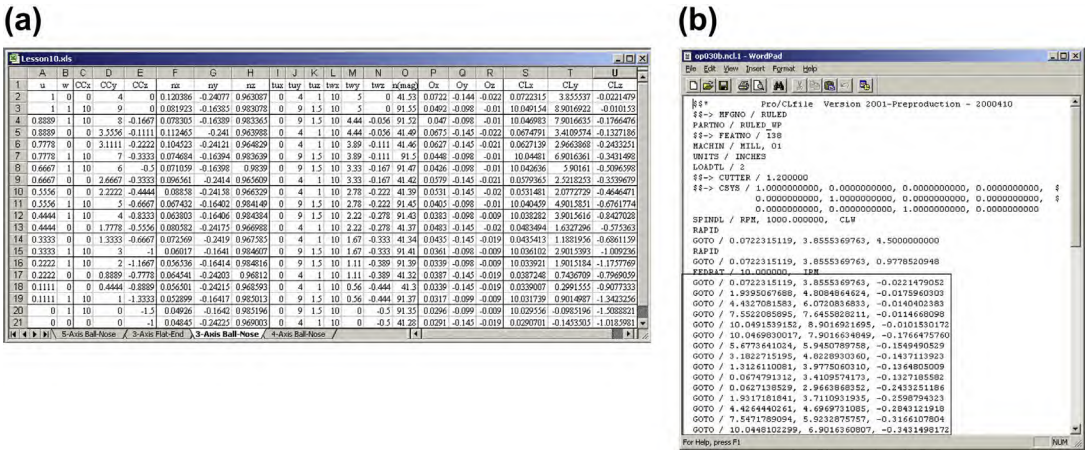


FIGURE 12.25

Toolpath verification for *OP030*, ball-nose cutter: (a) spreadsheet calculation and (b) CL data file generated by Pro/MFG.

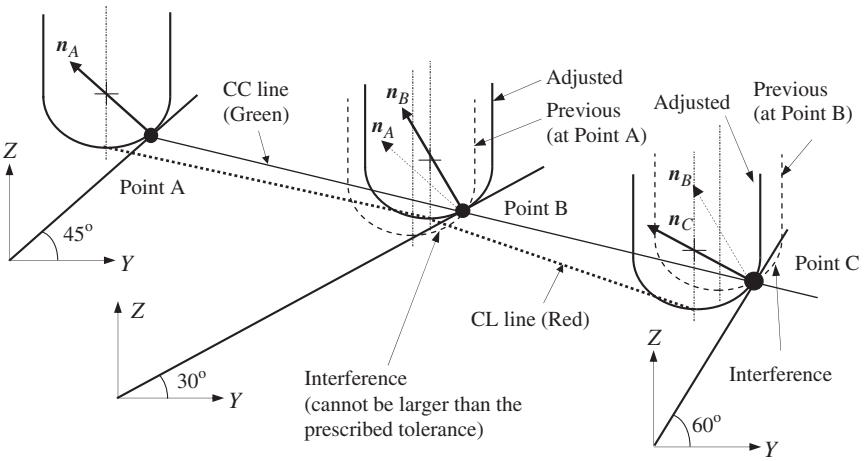


FIGURE 12.26

Illustration of cutter adjustment along a given pass. This figure is reproduced in color on the book's companion website: http://booksite.elsevier.com/9780123820389.

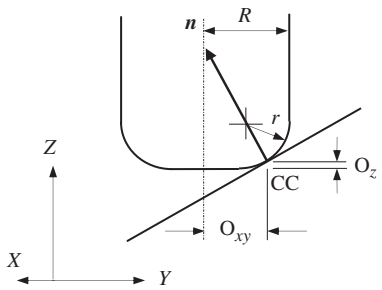


FIGURE 12.27

Illustration of cutter offset for a cutter with a small corner radius.

the cutter, based on the principle discussed earlier. Therefore, the normal vector passes through the center point of the circular arc of the corner round. In this case, the offsets can be calculated as follows,

$$O_{xy} = R - r\left(1 - \sqrt{n_x^2 + n_y^2}\right) \tag{12.24}$$

and

$$\begin{cases} O_x = O_{xy} \dfrac{n_x}{\sqrt{n_x^2 + n_y^2}} \\ O_y = O_{xy} \dfrac{n_y}{\sqrt{n_x^2 + n_y^2}} \\ O_z = -r(1 - n_z) \end{cases} \tag{12.25}$$

Now, let us discuss Question 2: Are these five CL data points collinear?

It is obvious that the CL data are not collinear, as illustrated in Figure 12.26, since the cutter offsets are not identical along the pass. The scallop-height calculation is identical to that of Section 12.2, using Eqs 12.4 and 12.6.

12.3.4 4-AXIS MILL WITH FLAT-END CUTTER* (OP020)

In this subsection, we will employ a 4-axis mill to conduct the surface milling for the same ruled surface of the block example. We choose the ball-nose cutter as defined in *OP020* and *NC Sequence 1: Contour Surface Milling*. Note that the front surface (which is normal to *X*-axis of *CS*1) of the block is chosen to define the rotational axis of the fourth axis. Thus, the cutter can only rotate along the *Y*-axis (in Pro/MFG). Consequently, the CL data will contain zero value for the *X*-component of the normalized vector that represents cutter orientation. The toolpath generated by Pro/MFG is shown in Figure 12.28(a) (top view).

Note that this toolpath is different from that of operations *OP010* and *OP030*. It is obvious since in *OP010* (5-axis mill with ball-nose cutter) the CL data contain all three components of the cutter

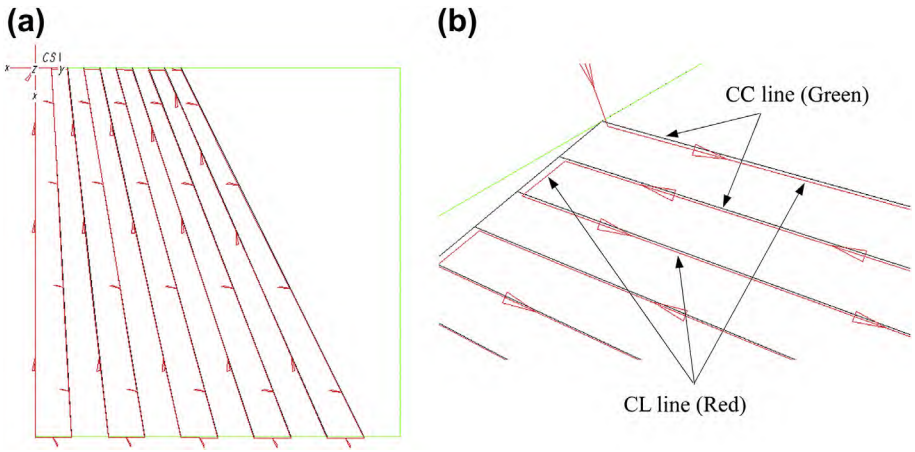


FIGURE 12.28

Toolpath of OP020: 4-axis mill: (a) top view and (b) closer view. This figure is reproduced in color on the book's companion website: http://booksite.elsevier.com/9780123820389.

orientation vector. In *OP030* (3-axis mill), no cutter orientation vector is needed. In addition, for a 5-axis mill, the cutter orientation must be adjusted at each CL data point to reduce excessive cuts. For a 3-axis mill, the cutter must be offset from CC line to maintain tangency between the cutter corner contour and the CC lines on the design surface.

Do CC and CL lines coincide in this operation with a 4-axis mill? Even though they are very close, as shown in Figure 12.28(b) in a closer view, they do not coincide.

12.3.4.1 CL Data

As discussed earlier, the 5-axis ball-nose cutter is normal to the design surface. In the current operation *OP020*, the cutter is free to rotate along the *Y*-axis only. At the same time, the basic principle—maintaining tangency between the cutter and the CC line—must be satisfied, as illustrated in Figure 12.29(a) and (b).

The CL data of the 4-axis mill must be offset from those of the 5-axis mill by compensating the *X*-component of the cutter orientation vector, at the same time maintaining tangency between cutter corner contour and CC lines. As shown in Figure 12.29(a), the cutter must rotate along the *Y*-axis with an amount that compensates the *X*-component of the cutter orientation vector. Consequently, the CL point will slide along the *X*-direction, as shown in Figure 12.29(b), by rotating the cutter along the *Y*-axis at the center point of the cutter's ball-nose.

Therefore, the adjusted vector $\boldsymbol{n}'$, as illustrated in Figure 12.29(a), can be calculated from the normal vector $\boldsymbol{n}$ of the 5-axis ball-nose cutter as follows,

$$\boldsymbol{n}' = \frac{[n_y, n_z]}{\sqrt{n_y^2 + n_z^2}} \tag{12.26}$$

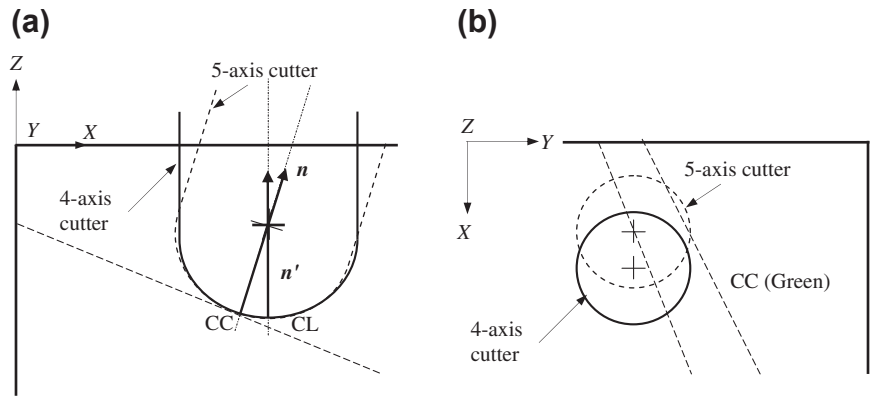


FIGURE 12.29

Cutter orientation: (a) side view and (b) top view.

Thus, the cutter offsets can be obtained as

$$\begin{aligned} O_x &= R\, n_x \\ O_y &= R\,(n_y - n'_y) \\ O_z &= R\,(n_z - n'_z) \end{aligned} \tag{12.27}$$

and the CL data can be obtained by using Eq. 12.23.

Note that part of the CL data calculated is listed in Figure 12.30(a) (Columns S to W of worksheet *4-Axis Ball-Nose* in the *Chapter 12.4.xls* file). They are identical to those generated by Pro/MFG (see Figure 12.30(b)).

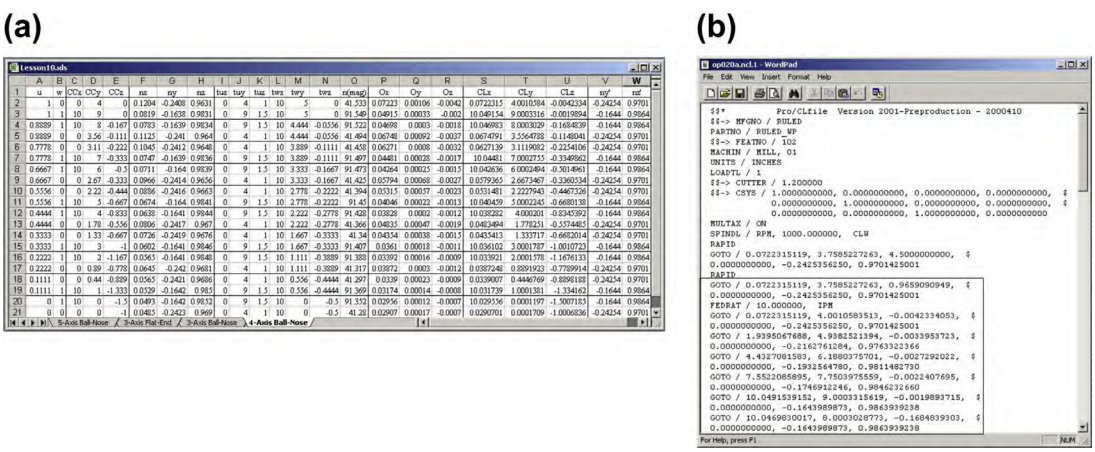


FIGURE 12.30

Toolpath verification for *OP020*: ball-nose cutter: (a) spreadsheet calculation and (b) CL data file generated by Pro/MFG.

12.4 CYLINDRICAL SURFACE OF BÉZIER CURVE

In this section, we discuss a surface milling sequence for a cylindrical surface formed by extruding a Bézier curve along the thickness direction (*Y*-direction of coordinate system *CS*0, chosen as machine zero), as shown in Figure 12.31(a). We will assume a 5-axis mill. This cylindrical surface is a special case of the ruled surface, where two path curves are identical and are parallel. The workpiece is a 10 in. × 5 in. × 4.5 in. block. A ball-nose cutter of 1.2 in. diameter and a stepover $S = 1.0$ in. are employed. The dimensions, or control point locations, of the Bézier curve are shown in Figure 12.31(b).

A toolpath of six identical passes is generated by Pro/MFG, as shown in Figure 12.32(a). There are 16 CL points generated along a given pass (in Pro/MFG). It can also be seen from Figure 12.32(b) that the machine surface is not quite smooth. We see tool marks due to the fact that the tool moves along a piecewise linear path instead of following the Bézier curve. How are these 16 CL points determined? Why not just, say, 5 CL points? How do we improve the surface finish?

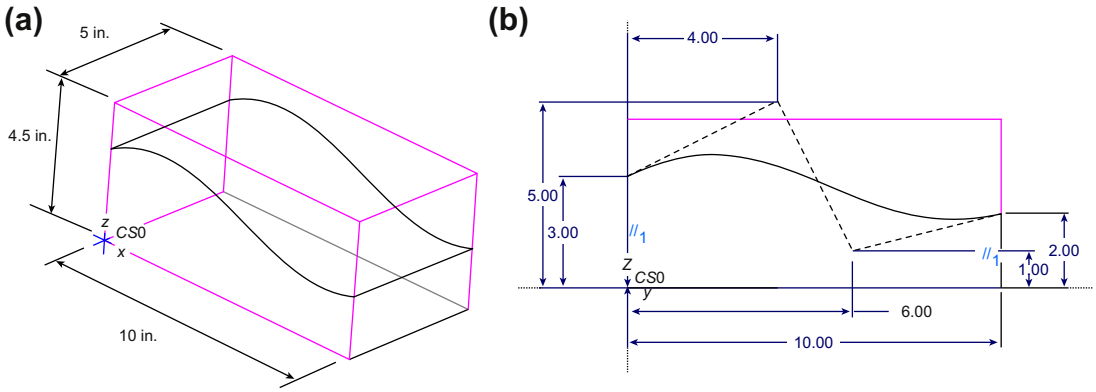


FIGURE 12.31

Contour surface milling for a cylindrical surface: (a) design model and (b) dimension of the Bézier curve.

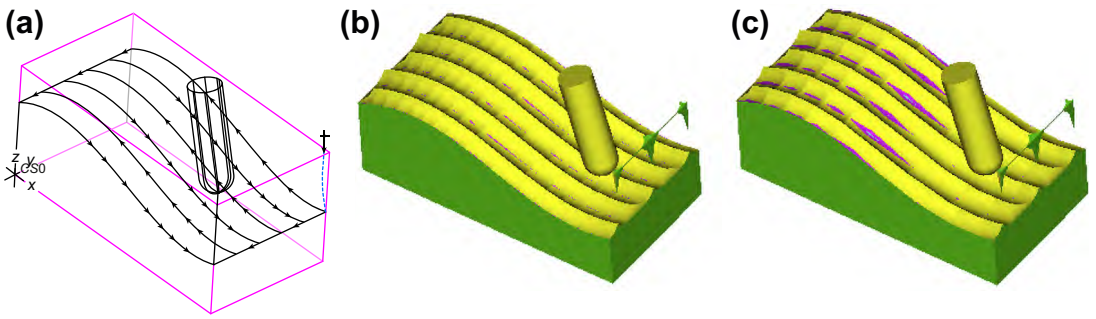


FIGURE 12.32

Contour surface milling for a cylindrical surface: (a) toolpath (tolerance = 0.01 in.), (b) machined surface (tolerance = 0.01 in.), and (c) machined surface (tolerance = 0.1 in.).

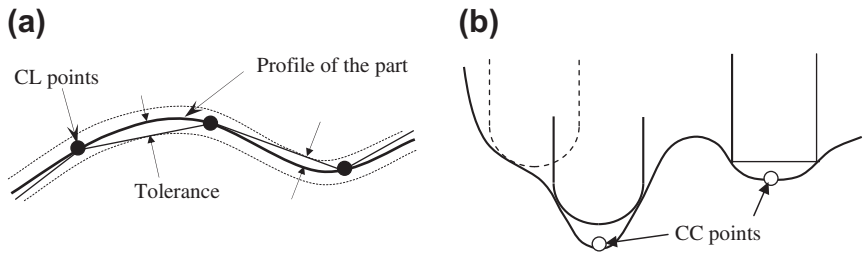


FIGURE 12.33

Toolpath generation: (a) influence of tolerance on CL points and (b) CC points not reachable.

The key factor is tolerance. A tolerance of 0.01 in. is prescribed for this example. The maximum distance between the CC curve and the line segments formed by two consecutive CL points must be smaller than this prescribed tolerance, resulting in 16 CL points. It is apparent that five CL data points along one pass will not satisfy the tolerance requirement.

What will happen if we change the tolerance from 0.01 to 0.1? Certainly, the number of CL points will be reduced, resulting in a rougher surface, as shown in Figure 12.32(c). The influence of tolerance on the final toolpath (in terms of CL data points) is illustrated in Figure 12.33(a). Can toolpath be generated all the time? The answer is no. When the CC points along the pass cannot be reached within a given tolerance (e.g., Figure 12.33(b)), no toolpath can be generated.

To illustrate the point, if we use a flat-end cutter, the toolpath generated in Pro/MFG only covers about half of the cylindrical surface, as shown in Figure 12.34(a). As a result, half of the surface will remain uncut, as shown in Figure 12.34(b). Why is only a partial toolpath (CL) generated? This is because when machining a concave surface, the cutter radius must be small enough so that the cutter tip (where CL data is defined) is able to reach the CC lines on the design surface. However, this is not always possible. When the cutter tip is not able to reach the CC lines on the design surface, no toolpath can be generated.

What can we do to force Pro/MFG to generate a complete path? We can certainly increase (relax) the tolerance value. If we increase tolerance from 0.01 to 0.5 in., the toolpath and the machined surface

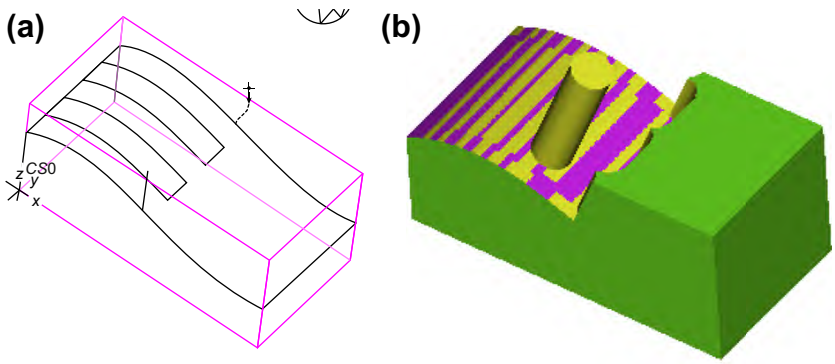


FIGURE 12.34

Surface milling for tolerance = 0.01 in. using a flat-end cutter: (a) toolpath and (b) machining simulation.

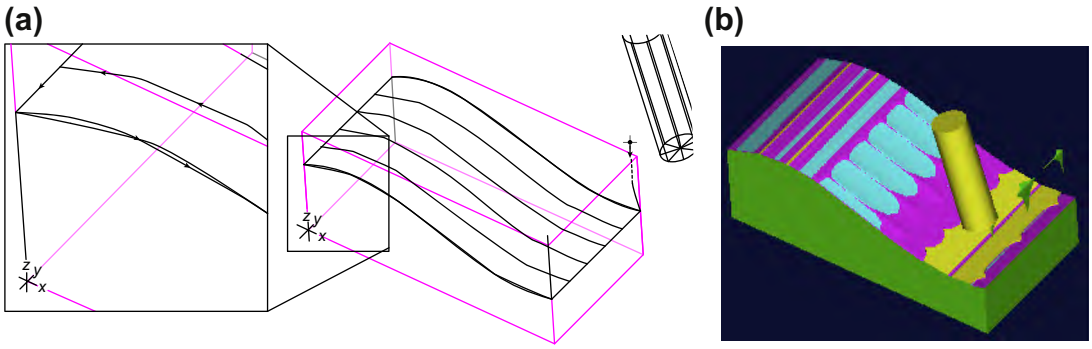


FIGURE 12.35

Surface milling for tolerance = 0.5 in. using a flat-end cutter: (a) toolpath and (b) machining simulation.

become the ones shown in Figures 12.35(a) and (b), respectively. A toolpath covers the entire design surface but with significant gouging (as illustrated in Figure 12.36(a)).

In such cases, there is a risk of the cutter interfering with or gouging the surface (Figure 12.36). Note that the interference could occur even with a 5-axis machine and a ball-nose cutter. To minimize the possibility of cutter plunging into the design surface resulting in an excessive cut, it is extremely important to go over the gouge checking in virtual machining.

Most virtual machining software, such as Pro/MFG, offer gouge-checking capability, in which locations on the design surface where an excessive cut that is larger than the allowed tolerance will be identified. For the surface shown in Figure 12.35(b), the toolpath was generated with a tolerance relaxed to 0.5 in. If the tolerance requirement is set back to 0.1 in., then a number of gouge points are identified by Pro/MFG, as shown in Figure 12.37.

In general, the accuracy of the machined surface is controlled by tolerance. On the other hand, the quality of the surface finish is controlled by the prescribed scallop height. Gouge-checking checks the interference between the cutter and design surface. Gouging affects both accuracy (gouge distance) and surface finish, which must be minimized.

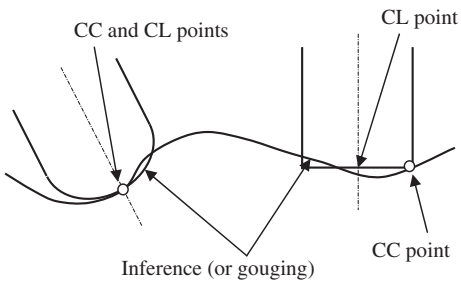


FIGURE 12.36

Illustration of surface gouging in machining.

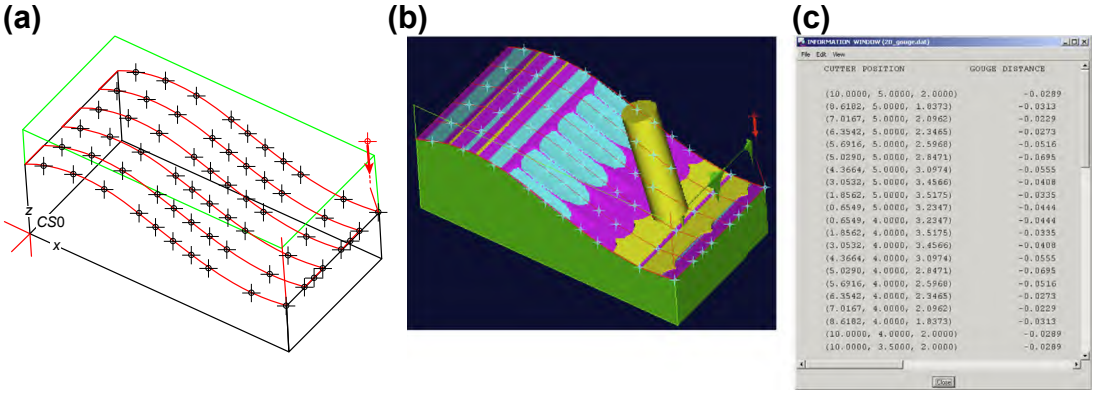


FIGURE 12.37

Gouging points identified on the design surface: (a) shown with toolpath, (b) shown with machining simulation, and (c) listed in a text file.

12.5 SUMMARY

In this chapter, we discussed the subject of toolpath generation. We used three examples of surface milling to illustrate toolpath calculations, including CL data and scallop height. With knowledge in the CL data calculation, you should be able to diagnose problems encountered in surface milling, especially gouging. Also, understanding scallop height calculation should help you become more confident in using virtual machining software to generate toolpaths, resulting in a quality machined surface.

With the discussion in Chapter 11 and this one, you should have a very good understanding in virtual machining and toolpath generation, which should help you address potential manufacturing issues encountered in product design; in addition, it is important to bring up these issues for consideration in the early product development stage.

In this next chapter, we will introduce another virtual manufacturing technology, sheet metal forming simulation, which has advanced significantly during the past two decades. Such forming technology has been widely used in the automotive industry to support body panel design and manufacturing, as well as in the aerospace industry in support of cowling and fuselage skin panel development.

QUESTIONS AND EXERCISES

12.1. The following ruled surface is formed by three straight lines and a quadratic Bézier curve (not drawn to scale). Corner points of the surface and control points of the Bézier curve are listed as follows:

$$\boldsymbol{p}_1 = [0,0,0], \boldsymbol{p}_2 = [1,0,1], \boldsymbol{p}_3 = [2,0,0]$$

$$\boldsymbol{p}_4 = [0,1,0], \boldsymbol{p}_5 = [2,1,0]$$

Note that (u,w) ∈ [0,1] × [0,1], and the parametric equation of the Bézier curve is

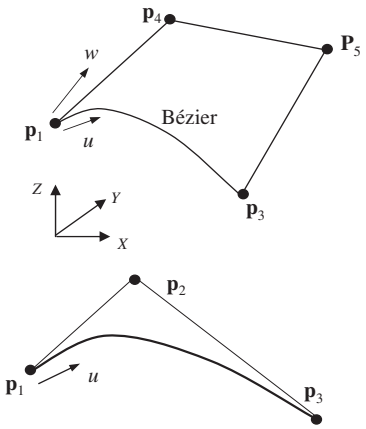
$$\boldsymbol{p}(u) = \boldsymbol{U}\ \boldsymbol{Q}\ \boldsymbol{G}$$

$$= \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \end{bmatrix}$$

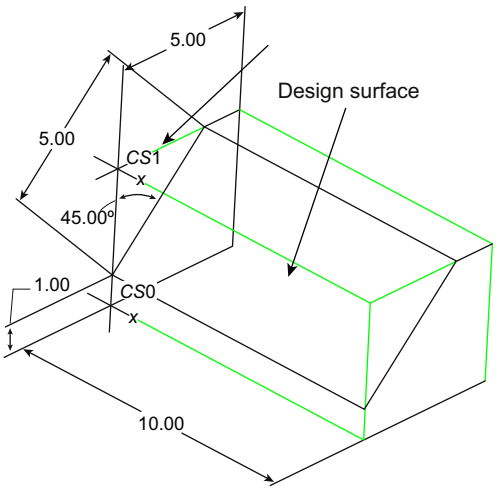
a. Compute CL data for a 5-axis milling machine with a ball-nose cutter of diameter 1.0 at $(u,w) = (0.5,0.5)$.

b. Will intermediate CL points between $(u,w) = (0.5,0)$ and $(u,w) = (0.5, 0.5)$ be needed if the tolerance is set to 0.01? State your rationale.

c. Will the number of CL data points be different along the passes of $w = 0$ and $w = 1$? State your rationale.

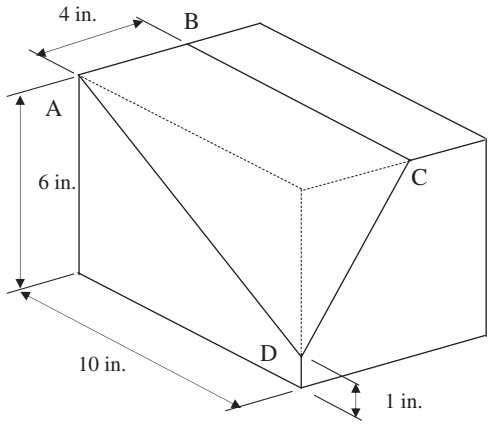


12.2. An inclined flat surface shown in the figure below is being machined using a 3-axis mill and a cutter of diameter 1.25 in. with a corner radius 0.125 in. If the toolpath is along the *X*-direction and if the stepover is 1 in., what will be the scallop height remaining on the design surface after machining?



12.3. A 5-axis mill is employed to machine the surface ABCD, as shown in the figure (next page), using a ball-nose cutter of diameter 1.2 in. and surface milling. If the number of passes $N = 10$ is specified, what is the maximum scallop height? If the scallop height is specified as 0.01 in., how many passes are required?

If a 3-axis mill is employed to machine the same surface using the same ball-nose cutter, and if the number of passes $N = 10$ is specified, calculate the maximum scallop height. If the scallop height is specified as 0.01 in., how many passes are required?



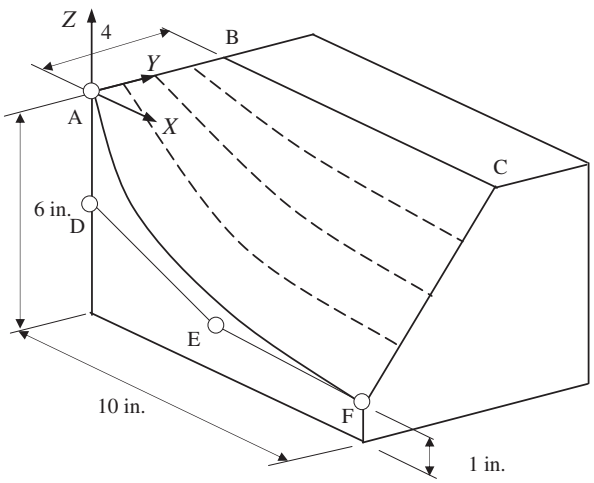
12.4. Generate a toolpath for surface ABCD, shown in the figure of Problem 3, using surface milling sequences for the following cases:

a. 5-axis ball-nose cutter of diameter 1.2 in.
b. 5-axis flat-end cutter of diameter 1.2 in.
c. 3-axis ball-nose cutter of diameter 1.2 in.
d. 3-axis flat-end cutter of diameter 1.2 in.

For all cases, assume that the number of passes is 5 and the tolerance is 0.001.
For each case, calculate CL data at points A, B, C, and D.

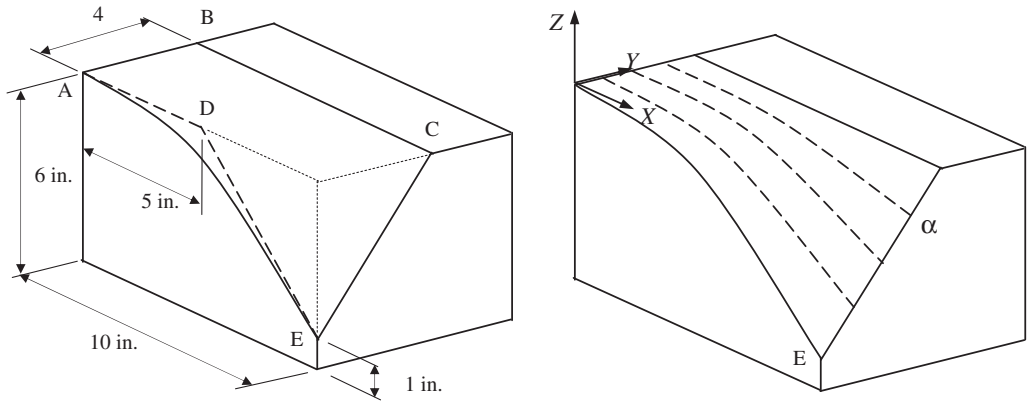
12.5. A 3-axis mill is employed to machine the surface ABCF, as shown in the figure, using a ball-nose cutter of diameter 1.5 in. Note that the curve AF is a cubic Bézier curve with four control points A (0,0,0), D (0,0,−3), E (5,0,−5), and F (10,0, −5).

a. If the number of passes $N = 5$ is specified, what will be the maximum scallop height?
b. If the maximum scallop height is defined as 0.05 in., and if the curve ADEF becomes a straight line from A to F, how many passes are needed?



12.6. A surface ABCE, shown in the figure, is being machined using a cutter of diameter 1.25 in. Note that the curve AE is a quadratic Bézier curve with three control points: A, D, and E.

- **a.** Derive parametric equations for the surface ABCE.
  Note that there are four CC curves evenly spaced along line AB (i.e., 1 in. apart). Calculate the CL data for the following cases.
- **b.** 5-axis mill with a ball-nose cutter of diameter 1.25 in. at point α
- **c.** 5-axis mill with a flat-end cutter of diameter 1.25 in. at point E
- **d.** 3-axis mill with a ball-nose cutter of diameter 1.25 in. at point α
- **e.** 3-axis mill with a flat-end cutter of diameter 1.25 in. at point E



REFERENCES

Choi, B., Jerrard, R., 1998. Sculptured Surface Machining: Theory and Applications, first ed. Kluwer Academic Publishers.

Choi, B.K., Kim, D.H., Jerrard, R.B., 1997. C-space approach to toolpath generation for die and mold machining. Computer Aided Design 29 (9), 657–669.

Kim, B.H., Choi, B., 2000. Guide surface based toolpath generation in 3-axis milling: An extension of guide plane method. Computer Aided Design 32, 191–199.

Kim, K.I., Kim, K., 1995. A new machine strategy for sculptured surfaces using offset surface. International Journal of Production Research 33 (6), 1583–1697.

Misra, D., Sundararajan, V., Wright, P.K., 2005. Zig-zag toolpath generation for sculptured surface finishing. Dimacs Series in Discrete Mathematics and Theoretical Computer Science 67, 265–280.

Sarma, R., Dutta, D., 1997. An integrated system for NC machining of multi-patch surfaces. Computer-Aided Design 29 (11), 741–749.

Sata, T., Kimura, F., Okada, N., Hosaka, M., 1981. A new method of nc interpolator for machining the sculptured surface. Annals of the CIRP 30 (1), 369–373.

CHAPTER

SHEET METAL FORMING SIMULATION

13

CHAPTER OUTLINE

13.1 Introduction 687
13.2 Fundamentals of Sheet Metal Forming 689
13.2.1 Sheet Forming Processes 689
13.2.1.1 Draw Forming 689
13.2.1.2 Stretch Forming 690
13.2.1.3 Sheet Hydroforming 691
13.2.2 Plane Stress and Material Properties 692
13.2.2.1 Stress–Strain Curve 692
13.2.2.2 Plane Stress Sheet Deformation 693
13.2.2.3 Material Anisotropy 697
13.2.3 Yield Criteria 698
13.2.3.1 Isotropic Yield Criteria 699
13.2.3.2 Anisotropic Yield Criteria 700
13.2.4 Forming Limit Diagram 701
13.2.5 Springback Analysis 704
13.2.6 Numerical Implementations 707
13.3 Process Planning and Tooling Design 709
13.3.1 One-Step Simulation for Formability Study 711
13.3.1.1 Blank Fitting and Blank Nesting 713
13.3.2 Die Design 714
13.3.2.1 Part Preparation 715
13.3.2.2 Binder Design 716
13.3.2.3 Addendum Design 716
13.3.2.4 Drawbead Design 718
13.3.3 Incremental Forming Analysis 719
13.3.4 Springback Analysis and Die Compensation 721
13.4 Commercial Forming Simulation Software 725
13.4.1 Overview of Simulation Software 725
13.4.1.1 FastForm 726
13.4.1.2 AutoForm 726
13.4.1.3 Pam-Stamp 2G 726

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00013-2
Copyright © 2015 Elsevier Inc. All rights reserved.

13.4.2 HyperForm 727
13.4.3 DynaForm 727
13.5 Case Studies 730
13.5.1 Core Panel 731
13.5.2 Wheel Fairing 734
13.6 Summary 739
Questions and Exercises 739
References 740

Sheet metal forming is one of the most important manufacturing processes for mass production, especially in the automotive and aerospace industries. Although a substantial resource is devoted to equipment purchase and initial setup, it is often cost-effective for large-quantity production.

In general, a great deal of effort is devoted to the design and production of reliable tooling, in which extensive knowledge and experience are required for design engineers to come up with tooling and process for sheet metal forming. It often takes trial-and-error measures to arrive at acceptable tooling and process that produce working parts. This is because a forming window in principal strain space, which identifies the strains that can be developed safely in a sheet element, is not readily available and is very much dependent on part geometry, tooling, and a forming process. Such a forming window is bounded by failure limits corresponding to localized necking, shear fracture, and wrinkling. Searching for a robust forming process that offers the strains in the part lie well within the forming window through a trial-and-error method is inefficient and expensive to say the least. Simulation technology and software tools offer engineers an effective alternative to achieve tooling design and a process that can produce quality parts more efficiently and with less cost.

Since the mid-1970s, important advances have been made in the development of computer simulation technology and software tools for deformation modeling during sheet metal forming. Computer simulation has been proved to be effective, and it shows promise in realistically simulating the sheet metal forming process and supporting tooling design.

Sheet metal forming simulation is a substantial subject that involves a broad range of topics in mechanics, numerical computations, and modeling and tooling design. Instead of providing a thorough and in-depth review on these topics, we will briefly discuss the mechanics and computational aspects of the forming simulation, and focus more on the simulation modeling and tooling design. Fundamental topics of mechanics in sheet forming (e.g., plastic behavior of sheet, formability, and springback) are briefly reviewed to provide readers some basic concepts in forming. For those who are interested in learning more about metal forming mechanics, you can refer to excellent books such as Marciniak et al. (2002) and Banabic (2010).

A computational aspect of the sheet metal forming using finite element analysis (FEA) will be briefly discussed to offer readers a basic understanding in the concepts and key elements involved in performing numerical analysis for forming simulations. We hope this brief introduction will provide some insight into numerical computations; thus, not necessarily viewing software tools as a complete black box. Those who want to dig more into computational theory and numerical implementation can refer to excellent references such as Wagoner and Chenot (2001). In addition to simulation, practical aspects of implementing scenarios assumed in simulation on the shop floor is no

small issue. Only a handful of articles can be found that deal with such issues, including Altan and Tekkaya (2012). This chapter is organized with the assumption that readers have a basic understanding of and knowledge about the sheet forming process and have not had experience in simulating processes using computer tools. Before going over this chapter, readers are strongly encouraged to review basic metal forming subjects in popular manufacturing process books such as Kalpakjian and Schmid (2010).

With a basic understanding of sheet metal forming mechanics and analysis, we devote a major effort to discussing sheet forming process planning and tooling design using simulation technology and software tools. We include one step simulation for formability study that is often an important first step in determining whether part design can be successfully formed. We discuss die design with addendum, and drawbead design that supports wrinkle reduction or removal if applied properly. We also include incremental forming analysis that shows the detailed steps in how a blank deforms during the forming process. Springback analysis, which plays an important role in compensating die geometry, so that the blank formed more closely matches the final part shape, is also included in the discussion.

There are tremendous advantages to using simulation technology and software tools to support sheet forming, tooling design, and process design. They help reduce product development time, enhance product quality, and reduce costs. Above all, the biggest advantage is that the simulation offers engineers visualizations of the blank deformation during the forming process with a display of engineering information (e.g., stress, strain, thickness distribution, and so on) over the blank sheet. Such rich technical information opens a door for engineers to understand exactly how the part is formed, how the blank is stretched and bent, and how the material flows during forming. Visualizing the blank deformation greatly enhances our understanding of the mechanics and physics in sheet metal forming; enables us to diagnose issues revealed in part design, tooling, or process; and makes us more competent in solving complex design and manufacturing issues in metal forming.

In addition to discussing forming simulation, we offer a review of major forming simulation software packages, including DynaForm, HyperForm, and so on. Such a review should provide readers with a general understanding about software availability and technical capabilities and provide enough information to make adequate software selection decisions when offered an opportunity. We also offer more insight into using simulation software for support of forming simulation through two case studies. Overall the objectives of this chapter are to provide a brief introduction to the basic mechanics and computational methods to help readers understand how the sheet forming simulation is carried out; to help readers become familiar with forming modeling and simulation so that the tools can be effectively used for design; to familiarize readers with existing commercial software; to support readers in learning how technology and software tools can be put into use for practical applications through case studies.

13.1 INTRODUCTION

In sheet metal forming, for example, a draw forming shown in Figure 13.1, a sheet (also called blank) is clamped around the edge and formed into a cavity by a punch. The metal is stretched by membrane forces and bent by bending moments so that it conforms to the shape of the tools. The membrane

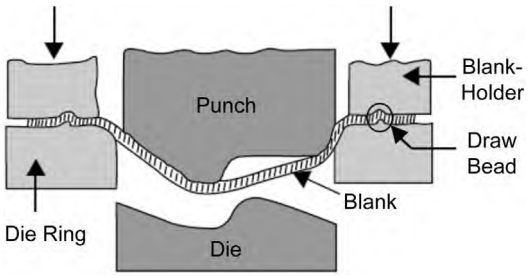


FIGURE 13.1

Schematic illustration of a draw forming process.

stresses in the sheet far exceed the contact stresses between the tools and the sheet, and the through thickness stresses may be neglected except at small tool radii. In general, the edge or flange is not held rigidly but is allowed to move inward in a controlled fashion. Usually, a blank holder is holding the blank by a binder force that provides tension sufficient to prevent wrinkling, but not enough to cause tearing. For some designs, drawbeads are added locally to minimize or eliminate wrinkles.

In general, a forming window in principal strain space that identifies the strains that can be developed safely in a material element is not readily available and is very much dependent on part geometry, tooling, and forming process. Such a forming window revealed in a graph of major and minor strains, called a forming limit diagram (FLD), is bounded by failure limits corresponding to localized necking, shear fracture, and wrinkle. It requires extensive knowledge and experience of design engineers to come up with a tooling and robust forming process so that the strains in the part lie well within the forming window. Even for designers with extensive knowledge and experience, achieving such a forming window is not intuitive and, on many occasions, requires several design iterations that involve adjusting tooling design or process parameters (e.g., binder force, blank size and shape, drawbead design, friction and lubricants). Going over design iterations by the trial-and-error method on the shop floor is extremely inefficient let alone the highly expensive tooling and labor costs. Sheet forming simulation offers engineers an alternative for achieving a tooling design and process that produce quality parts more efficiently and that are less expensive.

Since the mid-1970s, important advances have been made in the development of computer simulation technology and software tools for deformation modeling during sheet metal forming using finite element methods. Computer simulation has been proved to be effective and it shows promise in simulating the sheet metal forming process with excellent accuracy, therefore, adequately supporting tooling design for sheet forming manufacturing. In addition to forming simulation, blank design and nesting, addendum design, springback analysis, and die compensation are commonly supported by simulation software tools. Such simulation technology and software tools have been accepted and widely employed for part production in the automotive and aerospace industries.

This chapter introduces major topics in metal forming simulations, as well as serves as a gateway for readers to enter one of the extremely interesting and widely accepted areas of computer modeling and simulation. For readers to grasp basic concepts in forming simulation, we first provide (see Section 13.2) a review on the mechanics of sheet forming and numerical methods implemented in finite

element analysis (FEA) software. Following this brief discussion, we describe in Section 13.3 the chapter's main topic—applying the forming simulation technology to support tooling design and process planning. In Section 13.4, we provide an overview of commercially available simulation software tools. We point out a few advantages and shortfalls in terms of engineering capabilities offered by numerous tools, and discuss, in a bit more detail, the design capabilities offered by HyperForm (www.altair.com) and DynaForm (www.eta.com/DynaForm). Two case studies using practical examples are provided in Section 13.5 to illustrate a few more details about simulation, tooling design, and process planning. We also briefly mention the steps of using DynaForm software, one of the leading commercial softwares for sheet forming simulation.

13.2 FUNDAMENTALS OF SHEET METAL FORMING

In recent years, the application of simulation tools designed for the support of sheet forming manufacturing has increased considerably. These tools have been developed on top of the basic mechanics that describe the behavior of sheet metal material under various deformation conditions such as bending and stretching. A solid knowledge of the basic mechanics will be beneficial for engineers and designers to understand behind-the-scene operations in the software and to interpret and verify simulation results.

The aim of this section is to provide readers with a general understanding of the most fundamental theories of sheet metal forming from a mechanics perspective. We start with a brief introduction to the major sheet forming processes widely used in automotive and aerospace industries. Then in Sections 13.2.2 and 13.2.3, we focus on theoretical methods based on continuum mechanics that are important for modeling material behaviors and sheet deformation processes. We discuss in Section 13.2.4 the forming limit diagram as a useful tool for formability assessment. Basic mechanics for springback analysis also are briefly introduced in Section 13.2.5. Section 13.2 closes with a short introduction on numerical methods implemented in FEA for support of the simulation in Section 13.2.6.

13.2.1 SHEET FORMING PROCESSES

In a sheet forming process, a thin piece of metal sheet, commonly referred to as the blank, is bent or stretched by tools into a desired shape without excessive thinning, tearing, or wrinkling. Here we present a brief introduction to several sheet forming processes widely used in industry. Our focus will be on the three most frequent processes, namely draw forming, stretch forming, and sheet hydroforming. All three processes can be accurately simulated using most commercial sheet forming simulation software, which will be discussed later in Section 13.3.1.

13.2.1.1 Draw Forming

Draw forming (or deep draw) is one of the oldest and most widely used sheet forming processes. As shown in Figure 13.1, this process typically requires a punch, a die, and a blank holder as forming tools (the so-called tooling). During a drawing operation, the blank holder (or binder) clamps the blank over the binder surface of the die, while the punch moves toward the blank and squeezes it into the shaped die, forcing the material to deform plastically according to the geometry of the die cavity. The tooling motions in draw forming are in most cases hydraulically powered in order to deform the sheet with enough force, which can be as high as several hundred tons. After the process is completed, a trimming operation is usually required to remove unwanted blank material.

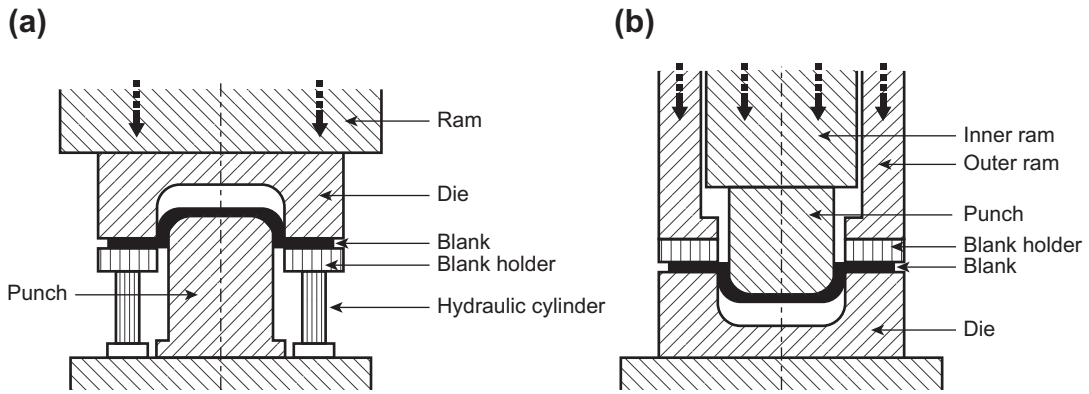


FIGURE 13.2

Schematic illustration of a draw forming process, (a) single-action draw, and (b) double-action draw.

Draw processes can be categorized based on the number of rams that can be operated independently of each other on the forming equipment. Schematic illustrations of the single-action draw (one ram) and double-action draw (two rams) are given in Figures 13.2(a) and (b), respectively. In a single-action draw operation, the female die is often the moving tool, and the blank holder force is provided by hydraulic cylinders (as depicted in Figure 13.2(a)) or a die cushion mechanism. The pressure of the cylinders needs to be adjusted during the process to control the force applied to the blank through the blank holder. For double-action draw, the outer and inner rams, which can be controlled independently, are used to drive the blank holder and the male punch, respectively. In both types, the blank holder force is a crucial parameter that controls the material flow and prevents excessive forming defects, especially wrinkling, in drawing.

Draw forming can be applied to the production of parts with a depth of more than half of its diameter. The process is most effective with ductile metals such as aluminum, brass, copper, and mild steel. Examples of parts formed with draw forming include automotive and aerospace body parts, kitchen hardware, sinks, and so on.

13.2.1.2 Stretch Forming

Stretch forming is mainly used for sheet metal parts with a large radius of curvature such as aircraft engine cowling. In this process, a piece of metal sheet with appreciable ductility is simultaneously stretched by tensile forces and bent over the tools. Instead of using a blank holder, the blank in stretch forming is tightly gripped along its edges by gripping jaws attached to hydraulically driven carriages. As illustrated in Figures 13.3(a) and (b), the metal sheet is first pulled by the grippers along the tensile direction, and then wrapped over the male block to the required final shape. In some cases, a female block may be used to form complex features on the part (Figure 13.3(c)). Note that Figure 13.3 shows a common stretch forming operation oriented in the vertical direction. Alternatively, in a horizontal stretch forming process (see, for example, Figure 7.30 of Kalpakjian and Schmid, 2010), the form die is mounted sideways on a stationary table, while the gripping jaws pull the sheet horizontally around the die to form the part.

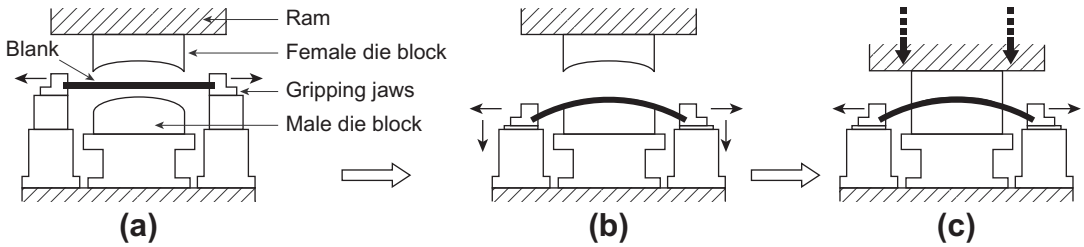


FIGURE 13.3

Schematic illustration of a vertical stretch forming process, (a) sheet pulled by the grippers along the tensile direction, (b) wrapped over the male block, and (c) female block to form complex geometric features.

Stretch forming is a quick, versatile, and economic production process that is capable of shaping metal sheets with very high accuracy and smooth surfaces. This process is ideally suited for the manufacturing of large curved parts made from aluminum, steel, and titanium. High accuracy, close and consistent tolerances, no surface marring, no distortion or ripples, and no surface misalignment of complex profiles are important inherent benefits of stretch forming. Moreover, the springback effect can be largely eliminated due to relatively uniform thinning and stress gradient on stretch formed parts.

13.2.1.3 Sheet Hydroforming

As an attractive alternative to conventional draw forming processes, sheet hydroforming (also called *fluid forming*) replaces one of the tooling blocks (punch or die) with hydraulic fluids to provide the shaping force. As illustrated in Figure 13.4, in sheet hydroforming the blank is formed into the shape of the male or female tool using fluid pressure applied through a rubber diaphragm; the fluid, together

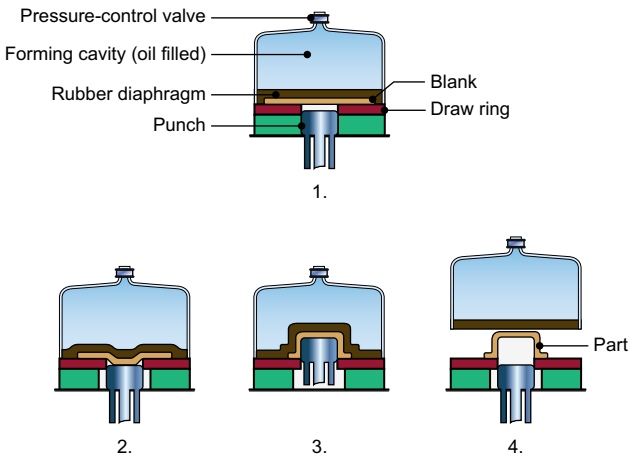


FIGURE 13.4

Schematic illustration of sheet hydroforming processes. (Figure 7.34, Manufacturing Processes for Engineering Materials, 5th ed., Kalpakjian Schmid, © 2008, Pearson Education ISBN No. 0-13-227271-7.)

with the rubber diaphragm, acts as a universal female–male tool. The maximum pressure for sheet hydroforming can be up to 100 MPa, and the allowed dimensions of the part depend on the chamber volume of the equipment.

Sheet hydroforming offers many advantages over traditional drawing operations. First, the setup procedure is simple, in which tooling can be quickly mounted and self-centered. No tooling fit is necessary since only one block is required, which also contributes to lower tooling cost. The flexible diaphragm minimizes and often eliminates draw marks normally created by matched die forming. Also, hydroforming flows the metal rather than stretching it, thus material thin-out can be kept to a minimum. Moreover, hydroforming is capable of forming parts with highly complex geometry, meanwhile keeping precise tolerances. The disadvantages of sheet hydroforming include relatively slower cycle time, expensive equipment, and so on.

In general, any sheet metal material capable of being cold formed, such as carbon steel, aluminum, stainless steel, copper, and brass, is a candidate for hydroforming. Unlike traditional forming operations, sheet hydroforming is an economical process for producing a relatively small number of parts.

13.2.2 PLANE STRESS AND MATERIAL PROPERTIES

When a material is deformed, two types of deformation occur: elastic and plastic. Elastic deformation is always the initial phase of loading, in which the material will change shape as load is applied. However, when the load is removed, the material returns to its original shape. The relationship between stress and strain in the elastic phase is usually linear for metals. In most metallic materials, when load is increased further, the elastic deformation will be accompanied by plastic deformation. In this load region, the material deforms not only elastically but also permanently. The stress–strain relationship in the elastic–plastic state is nonlinear because phenomena such as irreversible dislocation motions are the basis of plastic deformations. In metal forming, elastic strains are smaller by several orders of magnitude than plastic strains. Yet, in the case of sheet metal forming, they are still very much relevant since they are the reason for part springback.

13.2.2.1 Stress–Strain Curve

A simple uniaxial tensile test is widely used to measure and study many material properties in plastic regime. Consider a thin specimen with initial gauge length ℓ_0, width w_0, and thickness t_0, as illustrated in Figure 13.5(a). When the specimen is stretched by an axial load P, the gauge length, width, thickness, and the load P at any instant during the test can be recorded to create various material property diagrams. One of the most useful diagrams is the *true stress–strain curve* (Figure 13.5(b)). The true stress is defined as the current load P divided by the current cross-section area A of the specimen, as

$$\sigma = \frac{P}{A} \tag{13.1}$$

If the straining process continues uniformly, the true strain is given as an integral of strain increments along the load direction

$$\varepsilon = \int d\varepsilon = \int_{\ell_0}^{\ell} \frac{d\ell}{\ell} = \ln \frac{\ell}{\ell_0} \tag{13.2}$$

where ℓ represents the current gauge length.

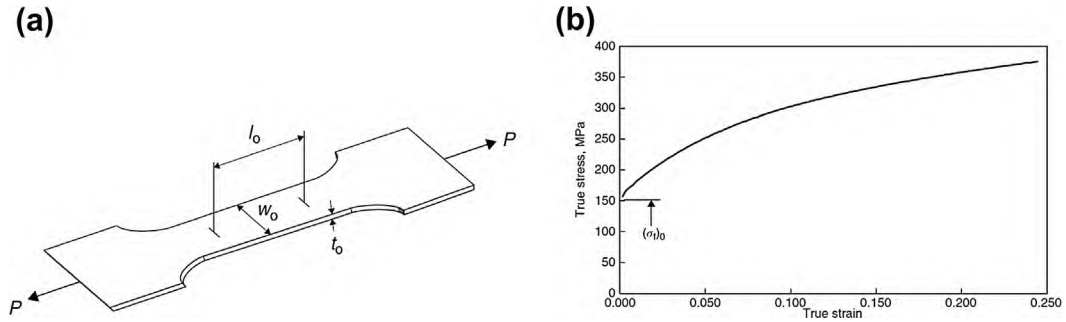


FIGURE 13.5

(a) A typical uniaxial tensile test strip, and (b) a true stress-strain curve (Marciniak et al., 2002).

In Figure 13.5(b), $(\sigma_f)_0$ indicates the initial flow stress. For isotropic materials, the flow stress is the stress at which the material would yield in simple tension; that is, the material yield strength with which we are familiar. In addition, it can be seen from Figure 13.5(b) that the yield stress increases as plastic deformation grows. This phenomenon is called *strain-hardening* (or work hardening), which is exhibited by most metals and alloys.

In a mechanics model, it is preferable to use a simple empirical law, often called *flow curve equation*, to approximate the stress–strain curve (or strain-hardening curve). One of the most common relations is obtained by fitting the experimental data with an equation in the form of

$$\sigma = K\varepsilon^n \tag{13.3}$$

where the exponent n is known as the *strain-hardening index* and K is usually referred to as the *strength coefficient*. This empirical equation, known as *power law* or Hollomon's law, is often used to describe the plastic properties of annealed low-carbon steel sheet (Marciniak et al., 2002). It provides an accurate description of the true stress–strain curve, except for the elastic regime and during the first few percent of plastic deformation, since this law predicts zero stress and an infinite curve slope at zero strain.

An alternative law, known as *Swift's law*, given by

$$\sigma = K(\varepsilon_0 + \varepsilon)^n \tag{13.4}$$

is also frequently used, where ε_0 is called the *pre-strain* or *offset strain*. A law in this form will fit a material with a definite yield strength. If the material has been hardened in some prior process, this constant ε_0 indicates a shift in the strain axis corresponding to this amount of strain (Marciniak et al., 2002). The two relations in Eqs 13.3 and 13.4 are illustrated in Figure 13.6 as solid curves, while the dotted line represents schematically the experimental curve.

In addition to Hollomon's and Swift's laws, several other flow curve equations (Palaniswamy and Billur 2012), such as Ludwik, Fields, and Backofen, are widely employed. Flow stress curves are very important in design of metal forming processes, since they describe material behavior during deformation.

13.2.2.2 Plane Stress Sheet Deformation

Forming a metal sheet to a given shape, for example, using a typical deep drawing shown in Figure 13.7(a), involves permanent plastic deformation of the material. In most sheet forming

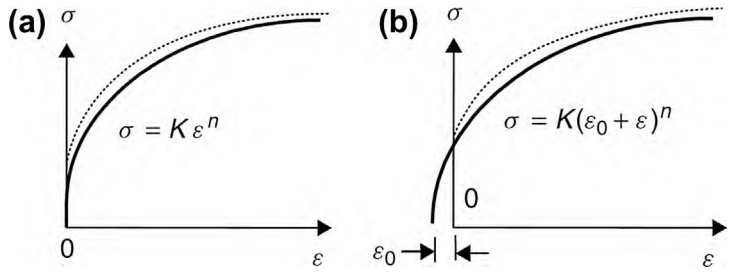


FIGURE 13.6

Empirical stress-strain laws fitted to an experimental curve, (a) power law, (b) power law with initial strain (Marciniak et al., 2002).

processes, the thickness of the blank is small compared to the in-plane dimensions. In addition, the stress perpendicular to the surface of the sheet is usually negligible because the contact pressure between the sheet and the tooling is generally much lower than the yield strength of the material, as illustrated in Figure 13.7(b). Therefore, the problem can be simplified by assuming the normal stress to be zero, resulting in a state of *plane stress* in which $\sigma_z = \tau_{xz} = \tau_{yz} = 0$. Note that z is usually the direction along the thickness, while x and y are in-plane coordinates. The out-of-plane shear strains, γ_{xz} and γ_{yz}, are also assumed to be zero for plane stress.

A monotonic and proportional deformation is usually assumed for the development of a simple theory for sheet forming. The term *monotonic* implies that the principal strain increments, $d\varepsilon$, increase smoothly in a constant direction without any inversion. A process is said to be proportional when the

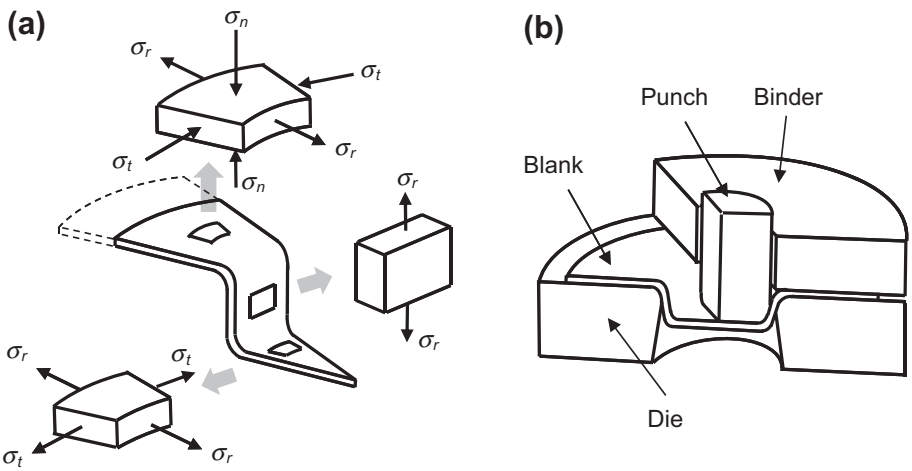


FIGURE 13.7

Schematic of deep drawing forming. (a) Various states of stress in deep drawing, and (b) deep draw process (Palaniswamy, 2012).

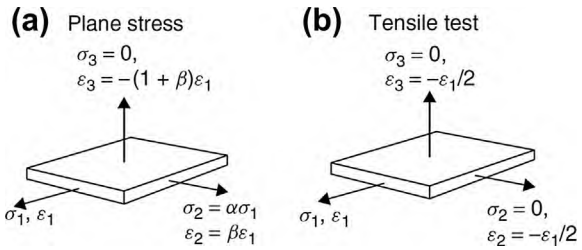


FIGURE 13.8

Principal stresses and strains for a material element deforming in (a) a general plane stress sheet process and (b) a uniaxial tensile test (Marciniak et al., 2002).

ratio of the principal strains remains constant from the onset of yielding to the maximum loading condition. In this case, it is convenient to describe the deformation of a material element in terms of strain ratio β and the stress ratio α (see Figure 13.8(a)) (Marciniak et al., 2002), as

$$\begin{aligned} &\text{Strain: } \varepsilon_1; \varepsilon_2 = \beta\varepsilon_1; \varepsilon_3 = -\varepsilon_1 - \varepsilon_2 = -(1+\beta)\varepsilon_1 \\ &\text{Stress: } \sigma_1; \sigma_2 = \alpha\sigma_1; \sigma_3 = 0 \end{aligned} \tag{13.5}$$

Note that the usual convention is to define the principal directions so that $\sigma_1 > \sigma_2$, while the third direction is perpendicular to the surface. The principal strain in the third direction is obtained based on the constant volume condition (i.e., $\varepsilon_1 + \varepsilon_2 + \varepsilon_3 = 0$). For uniaxial tension, the strain and stress ratios are $\beta = -1/2$ and $\alpha = 0$, respectively (Figure 13.8(b)).

According to the Levy–Mises flow rule (Levy, 1870), which states that the ratio of the strain increments will be the same as the ratio of the deviatoric stresses (that causes shear deformation, which is defined as the normal stresses reduced by the hydrostatic stresses), a relation between the stress and strain ratios can be derived as

$$\alpha = \frac{2\beta + 1}{2 + \beta} \quad \text{or} \quad \beta = \frac{2\alpha - 1}{2 - \alpha} \tag{13.6}$$

Within an increment in a process, the plastic work done in deforming a unit cube element is given by

$$\frac{dW}{vol.} = \sigma_1 d\varepsilon_1 + \sigma_2 d\varepsilon_2 + \sigma_3 d\varepsilon_3 \tag{13.7}$$

where *vol.* is the volume of the unit cube element. It is also usual to express Eq. 13.7 in the form of

$$\frac{dW}{vol.} = \sigma_e \cdot \varepsilon_e \tag{13.8}$$

where σ_e and ε_e are the *effective stress* (or equivalent stress) and *effective strain* (or equivalent strain), respectively. The expressions of the effective stress and strain based on the isotropic von Mises yield criterion—discussed in the next section—can be written for plane stress as

$$\sigma_e = \sqrt{1 - \alpha + \alpha^2} \cdot \sigma_1 \text{ and } \varepsilon_e = \sqrt{\frac{4}{3}\left(1 + \beta + \beta^2\right)} \cdot \varepsilon_1 \tag{13.9}$$

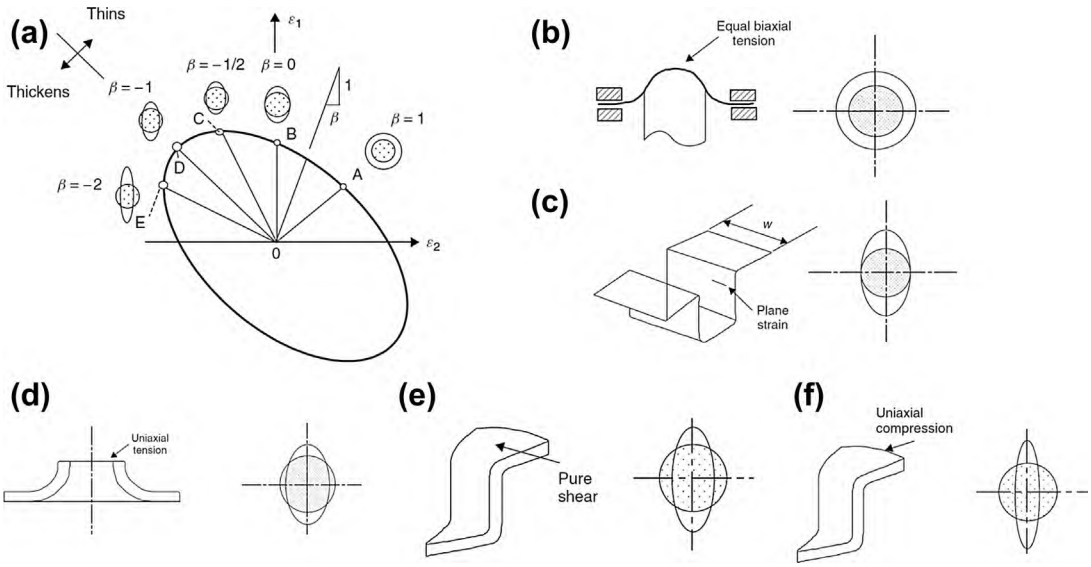


FIGURE 13.9

Schematic forming diagram and deformation modes, (a) the strain diagram showing the different deformation modes corresponding to different strain ratios. (b) Equibiaxial stretching at the pole of a stretched dome. (c) Deformation in plane strain in the side-wall of a long part. (d) Uniaxial extension of the edge of an extruded hole. (e) Drawing or pure shear in the flange of a deep-drawn cup, showing a grid circle expanding in one direction and contracting in the other. (f) Uniaxial compression at the edge of a deep-drawn cup (adapted with permission, Figure 3.3, Page 35, Marciniak et al., 2002).

For an isotropic material at yielding, the effective stress is equal to the flow stress (or instantaneous yield strength), and the effective stress–strain curve is coincident with the tensile test true stress–strain curve (Figure 13.5(b)). At any instant during a monotonic and proportional plane stress sheet deformation process, if the current principal strains are measured, the strain ratio β and the effective strain ε_e can be calculated. The stress state (i.e., σ_1 and σ_2) can then be determined using Eq. 13.9 with the stress ratio α calculated from Eq. 13.6, as well as the effective stress σ_e for the current effective strain found according to the strain-hardening curve.

Figure 13.9 illustrates the characteristics of the various modes of plain stress sheet deformation. The ellipse shown is a contour of equal effective strain ε_e. When we assign ε_1 to be the larger one of the principal strains, all points will be to the left of the right-hand diagonal (45° line). If a linear strain path is assumed, the five straight lines OA to OE indicate equal biaxial stretching ($\beta = 1$), plane strain ($\beta = 0$), uniaxial tension ($\beta = -1/2$), constant thickness drawing ($\beta = -1$), and uniaxial compression ($\beta = -2$), respectively. Note that according to Eq. 13.5, when $\beta > -1$, the sheet will get thinner ($\varepsilon_3 < 0$) during the process; conversely, for any point below the $\beta = -1$ line, the sheet becomes thicker.

EXAMPLE 13.1

Consider a square Aluminum 2024 sheet with edge length 1.5 in. that is uniformly stretched in two directions into a 1.58 in. ×1.74 in. rectangle. The sheet is initially 0.05 in. thick, and the material behavior is assumed to be isotropic and well fitted by a power law with parameters $K = 113$ *ksi* and $n = 0.17$. If the process is monotonic and proportional, calculate the principal stresses and sheet thickness at the end of the process.

Solution

The principal strains at the end of the process can be first evaluated as

$$\varepsilon_1 = \ln\frac{1.74\text{ in.}}{1.5\text{ in.}} = 0.148; \varepsilon_2 = \ln\frac{1.58\text{ in.}}{1.5\text{ in.}} = 0.0520$$

which gives the final thickness of the material; that is,

$$\varepsilon_3 = \ln\frac{t}{t_0}$$

and

$$t = t_0 e^{\varepsilon_3} = t_0 e^{-\varepsilon_1-\varepsilon_2} = 0.05\ \text{in.} \times e^{-0.148-0.052} = 0.041\text{ in.}$$

The strain ratio and effective strain can also be obtained as

$$\beta = \frac{\varepsilon_2}{\varepsilon_1} = \frac{0.052}{0.148} = 0.351;\ \ \varepsilon_e = \sqrt{\frac{4}{3}\left(1+\beta+\beta^2\right)}\cdot\ \ \varepsilon_1 = \sqrt{\frac{4}{3}(1+0.351+0.351^2)}\cdot 0.148 = 0.207$$

Then the effective stress can be calculated from the strain-hardening behavior of the material, as

$$\sigma_e = K{\varepsilon_e}^n = 113\ \ ksi \times 0.207^{0.17} = 86.5\ psi$$

Finally, according to the stress ratio

$$\alpha = \frac{2\beta+1}{2+\beta} = \frac{2\times 0.351+1}{2+0.351} = 0.724$$

And the principal stresses can be determined as

$$\sigma_1 = \frac{\sigma_e}{\sqrt{1-\alpha+\alpha^2}} = \frac{86.5\ psi}{\sqrt{1-0.724+0.724^2}} = 96.7\,psi; \sigma_2 = \alpha\sigma_1 = 0.724\times 96.7\,psi = 70.0\,psi$$

13.2.2.3 Material Anisotropy

It is well known that sheet metal materials in general exhibit significant anisotropy because of their crystallographic structure and the characteristics of the rolling process (Banabic, 2010). In fact, most metal sheets are orthotropic, since the mechanical properties are symmetric with respect to three orthogonal planes. The variation of the material behavior with direction can be assessed by a quantity named *Lankford parameter* or *anisotropy coefficient* (Lankford et al., 1950), which is given by

$$R = \frac{\varepsilon_w}{\varepsilon_t} \tag{13.10}$$

This coefficient is determined by uniaxial tensile tests (Figure 13.5(a)) on specimens in the form of a strip cut along different directions of the sheet, where ε_w and ε_t are the strains in the width and thickness directions of the specimen, respectively. The anisotropy coefficient can also be written as

$$R = \frac{\ln \frac{w}{w_0}}{\ln \frac{w_0 \ell_0}{w \ell}} \tag{13.11}$$

where $\varepsilon_t = \ln \frac{t}{t_0} = \ln \frac{w_0 \ell_0}{w \ell}$ to avoid the measurement of the strain along the thickness direction.

The direction in which these coefficients are measured can be indicated by a suffix (i.e., R_0, R_{45}, and R_{90}, respectively) for tests in the rolling, diagonal, and transverse directions. By convention, the anisotropy coefficients are usually determined at 20% elongation for the purpose of comparison (Banabic, 2010).

In some cases, it is convenient to consider all in-plane directions as equivalent, with only the thickness direction different. This condition is called *normal anisotropy*, in which case all anisotropy coefficients are assumed to be equal (Wagoner and Chenot, 2001). The *coefficient of normal anisotropy* (or normal anisotropy ratio) can be determined by averaging anisotropy coefficients measured in individual directions, as

$$\overline{R} = \frac{R_0 + 2R_{45} + R_{90}}{4} \tag{13.12}$$

On the other hand, to describe the variation of anisotropy coefficients within the sheet plane, a quantity given by

$$\Delta R = \frac{R_0 - 2R_{45} + R_{90}}{2} \tag{13.13}$$

is commonly used, known as the *coefficient of planar anisotropy.*

13.2.3 YIELD CRITERIA

Yield criteria define the condition for the limit of elastic behavior or the onset of plastic deformation in a material under multi-axial states of stress. A criterion used for determining the condition of continuing plastic flow is also called the *flow criterion.* In the uniaxial compression or tensile test, the metal starts to flow plastically when the stress in the material reaches the material's yield strength; that is,

$$\sigma = \frac{P}{A} = \sigma_f \tag{13.14}$$

where P is the instantaneous force, A is the instantaneous area of the test specimen, and σ_f is the yield strength of the material (called flow stress). In multi-axial states of stress, the determination of plastic flow is not trivial and depends on a combination of all the stresses in the stress tensor. In this case, an implicit function of stresses can be used to describe the stress state at which the material will deform plastically. This function, which is called the *yield function* (Banabic, 2010), can be expressed by

$$F(\sigma, \sigma_f) = 0 \tag{13.15}$$

All stress states that satisfy $F < 0$ are related to an elastic deformation of the material. When $F = 0$, the material undergoes plastic deformation. Note that $F > 0$ has no physical meaning.

There are a number of theories available for predicting the yield conditions for both isotropic and anisotropic materials. These theories are based on different hypotheses about material behavior, and result in yield functions of different forms.

13.2.3.1 Isotropic Yield Criteria

For isotropic materials, plastic yielding depends only on the magnitude of the principal stresses and not on their directions (Kobayashi et al., 1989). The most widely used isotropic yield models are the *Tresca criterion* (Tresca, 1864) and the *von Mises criterion* (or Maxwell–Huber–Mises criterion) (von Mises, 1913).

The Tresca yield criterion is proposed based on the assumption that yielding would occur when the greatest maximum shear stress reaches a critical value. In the general case, the criterion can be written as

$$\max\{|\sigma_1 - \sigma_2|,\ |\sigma_2 - \sigma_3|,\ |\sigma_3 - \sigma_1|\} - \sigma_f = 0 \tag{13.16}$$

where σ_1, σ_2, and σ_3 are principal stresses and σ_f is the flow stress. Under plane stress condition, Eq. 13.16 becomes

$$|\sigma_1 - \sigma_2| - \sigma_f = 0 \tag{13.17}$$

which represents a polygon in the plane of the principal stresses σ_1 and σ_2, as shown in Figure 13.10(a). This hexagon can be thought of as the locus of a point P that indicates the stress state at yield as the stress ratio α changes. For a strain-hardening material, this locus will expand as the flow stress σ_f increases.

The von Mises yield criterion states that yielding will occur when the elastic energy of distortion reaches a critical value. For plane stress, the criterion can be written in the form

$$\sqrt{\sigma_1^2 - \sigma_1\sigma_2 + \sigma_2^2} - \sigma_f = 0 \tag{13.18}$$

As illustrated in Figure 13.10(b), the yield locus corresponding to the von Mises criterion is an ellipse.

For isotropic materials, both the preceding criteria are sufficiently accurate for approximation. Although there are major differences in the mathematical form of these two criteria, the values of stress predicted for any given value of stress ratio α will not differ by more than 15% (Marciniak et al., 2002). However, the von Mises criterion is continuous and convenient to use in numerical analysis (Kobayashi et al., 1989).

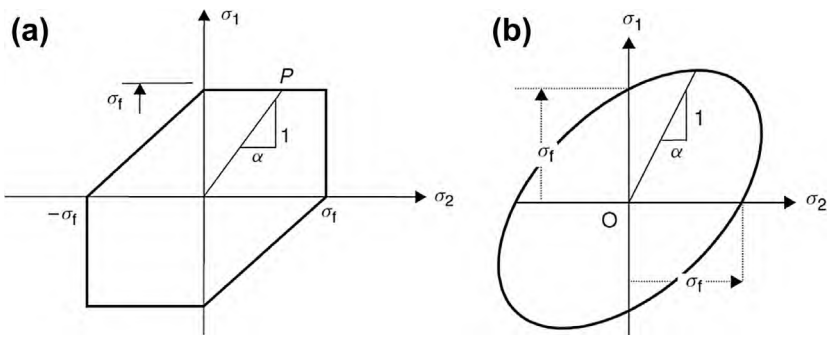


FIGURE 13.10

Plane stress yield loci for isotropic yield criteria, (a) the Tresca criterion, and (b) the von Mises criterion (Marciniak et al., 2002).

13.2.3.2 Anisotropic Yield Criteria

To accurately model sheet metal materials, which are supposed to have anisotropy with three orthogonal symmetry planes, the effect of anisotropy on the deformation characteristics needs to be considered. In 1948, Hill (1948) proposed an anisotropic yield criterion as a generalization of the von Mises model by assuming the yield function to be quadratic, as

$$f(\sigma_y - \sigma_z)^2 + g(\sigma_z - \sigma_x)^2 + h(\sigma_x - \sigma_y)^2 + 2l\tau_{yz}^2 + 2m\tau_{zx}^2 + 2n\tau_{xy}^2 - 1 = 0 \tag{13.19}$$

where f, g, h, l, m, and n are coefficients of material property. Note that the stress subscripts x, y, and z correspond, respectively, to the rolling, transverse, and thickness directions of the metal sheet. Assume that X, Y, and Z are the tensile yield strengths in the three principal directions of anisotropy, respectively; it can be obtained from Eq. 13.19 that

$$X^2 = \frac{1}{g+h},\ Y^2 = \frac{1}{f+h},\ Z^2 = \frac{1}{f+g} \tag{13.20}$$

From Eq. 13.20, constants f, g, and h can be obtained as functions of yield strengths X, Y, and Z in three orthogonal directions. Similarly, assuming R, S, and T are shear yield strengths in the three orthogonal directions, respectively, the remaining constants l, m, and n can be obtained as

$$R^2 = \frac{1}{2l},\quad S^2 = \frac{1}{2m},\quad T^2 = \frac{1}{2n} \tag{13.21}$$

Under a plane stress condition, Eq. 13.19 can be reduced to

$$(g+h)\sigma_x^2 - 2h\sigma_x\sigma_y + (h+f)\sigma_y^2 + 2n\tau_{xy}^2 - 1 = 0 \tag{13.22}$$

Note that while f, g, h, and n are not definite, their ratios are related to the anisotropy properties of the material; that is,

$$\frac{h}{g} = R_0,\quad \frac{h}{f} = R_{90},\quad \frac{n}{g} = \left(R_{45} + \frac{1}{2}\right)\left(1 + \frac{R_0}{R_{90}}\right) \tag{13.23}$$

where R_0, R_{45}, and R_{90} are anisotropy coefficients discussed in Section 13.2.2.

From Eqs 13.22 and 13.23, we have

$$h(\sigma_y - \sigma_x)^2 + \frac{h}{R_0}\sigma_x^2 = \frac{h}{R_{90}}\sigma_y^2 + h(2R_{45}+1)\left(R_{45} + \frac{1}{2}\right)\tau_{xy}^2 - 1 = 0 \tag{13.24}$$

Consider a tensile test (before necking) or compression test (before bulging), where the deformation is uniaxial along the rolling direction, the plastic deformation begins when Eq. 13.22 holds. Hill's yield function at the start of plastic deformation in uniaxial loading along the rolling direction can be obtained from Eq. 13.24 as:

$$h\left(1 + \frac{1}{R_0}\right)\sigma_f^2 - 1 = 0 \tag{13.25}$$

where σ_x reaches the flow stress. Combining Eqs 13.23 and 13.25, we get Hill's yield criterion in terms of constants that can be measured from experiments:

$$\sigma_f^2 = \frac{R_0R_{90}(\sigma_y - \sigma_x)^2 + R_{90}\sigma_x^2 + R_0\sigma_y^2 + (2R_{45}+1)(R_0+R_{90})\tau_{xy}^2}{R_{90}(R_0+1)} \tag{13.26}$$

Therefore, either Eq. 13.24 or 13.26 can be used to describe the behavior of sheet material in the finite element (FE) simulations by Hill's (1948) yield criterion. The Hill (1948) yield criterion is widely used in practice due to its simplicity. The basic assumptions are easy to understand, and the model has a simple formulation even for a 3D case. For plane stress, four parameters are sufficient to determine the yield function, namely R_0, R_{45}, R_{90}, and X (Banabic, 2010).

However, this quadratic yield criterion is inadequate to describe the plastic behavior of some materials such as aluminum alloys. In later work, Hill introduced several nonquadratic functions—for example, Hill (1979), Hill (1987), Hill (1990) and Hill (1993)—to extend the application of the yield criteria to various types of materials. The determination of these nonquadratic yield functions generally demands more material parameters.

Besides the family of Hill yield criteria, another class of models based on the isotropic formulation proposed by Hershey (1954) is also frequently used in modeling the yielding of sheet metal materials. Examples of these models are a series of yield functions developed by Barlat and coworkers (Barlat and Lian, 1989; Barlat et al., 1997a, 1997b, 2005). In 1989, Barlat and Lian (1989) proposed a yield criterion for materials exhibiting planar anisotropy, which takes the form

$$a|k_1 + k_2|^M + a|k_1 - k_2|^M + c|2k_2|^M = 2Z_x^M \tag{13.27}$$

where the exponent M is related to the crystallographic structure of the material. It is concluded in Logan and Hosford (1980) that the best approximation is achieved with $M = 6$ for BCC (body-centered cubic) materials and $M = 8$ for FCC (face-centered cubic) materials. In Eq. 13.27, k_1 and k_2 are invariants of the stress tensor given by

$$k_1 = \frac{\sigma_x + h\sigma_y}{2}; \quad k_2 = \left[\left(\frac{\sigma_x - h\sigma_y}{2}\right)^2 + p^2\tau_{xy}^2\right]^{1/2} \tag{13.28}$$

and a, c, h, and p are material parameters.

The Barlat and Lian (1989) criterion provides a good prediction of the yield locus for aluminum alloys without high anisotropy. Later extensions of this criterion have been reported (Barlat et al., 1997a, 1997b, 2005) to remove some of its drawbacks and limitations.

In addition to those mentioned before, many other anisotropic yield criteria have been developed by different research groups: the Cazacu-Barlat criterion (Cazacu and Barlat, 2001), the Vegter criterion (Vegter et al., 1995), and the BBC (Banabic-Balan-Comsa) criteria (Banabic et al., 2000). At present, the most frequently used yield criteria are Hill (1948), Hill (1990), and Barlat (1989) (Banabic, 2010). When choosing the yield criterion, several important factors (e.g., flexibility, degree of generality, number of mechanical parameters needed to determine the yield function, and the accuracy in predicting the yield locus must be taken into account. In addition, for numerical implementation, the computational efficiency of the yield functions needs to be considered. For example, the Hill (1990) criterion models plasticity locus in a more general manner than the Hill (1948) model but is much more computational expensive (ESI Group, 2011).

13.2.4 FORMING LIMIT DIAGRAM

The *forming limit diagram* (FLD), also known as the Keeler–Goodwin diagram, was originally derived as an experimental, semiquantitative tool to aid designers in evaluating the risks of local

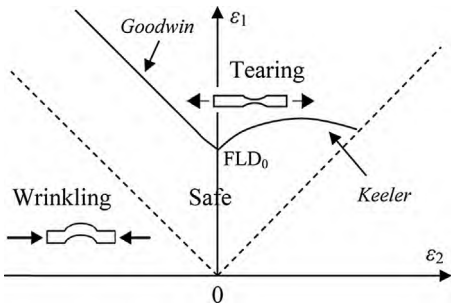


FIGURE 13.11

A schematic forming limit diagram. The strain diagram showing the different deformation modes corresponding to different strain ratios.

fracture and necking in sheet forming (Wagoner et al., 2001). It is now used frequently in failure diagnosis of sheet forming processes and has been implemented in most sheet forming simulation software.

As shown in Figure 13.11, a FLD is divided into different zones by several curves. The vertical and horizontal axes correspond to the major and minor strains, respectively. At any instant during a forming process, the strain at each location on the sheet is represented as a point in the forming limit diagram. The formability of the sheet can be evaluated by comparing the positions of these points to the curves.

The solid curve in Figure 13.11, which is called the *forming limit curve* (FLC), separates the safe and failure zones. A strain state above the FLC implies local necking or fracture. The left- and right-hand sides of the FLC were originally proposed by Goodwin (1968) and Keeler (1961), respectively, through mechanical tests. In practice, there is some scatter in the measured necking strains, and instead of a single curve, there is a band within which necking is likely to occur (Marciniak et al., 2002). The intersection of the forming limit curve with the vertical axis is noted as FLD_0, which depends mainly on the strain-hardening coefficient n and the thickness of the material (Banabic, 2010).

FLCs for mild steel, HSS (high-strength steel), and selective AHSS (advanced high-strength steel) have been studied, and the FLD_0 for the sheet material is approximated by the Keeler–Brazier equation based on the initial sheet thickness t_0 and the strain-hardening exponent n of the material (Palaniswamy and Billur, 2012)

$$FLD_0 = \ln\left[1 + (23.3 + 14.13t_0)\frac{n}{0.21}\right], \quad n \leq 0.21 \tag{13.29}$$

As discussed in the previous section, all possible strain coordinates will be above the dotted 45° line if ε_1 is always the larger principal strain component. In addition, the strain points located below the −45° dotted line indicate a thickening of the sheet, and thus a strong wrinkling tendency. However, in the FLC, the effect of strain as a result of bending is not included in establishing the limits. Therefore, the FLC must be used with caution in regions that undergo excessive bending (Palaniswamy and Billur, 2012).

Another important point to keep in mind is the factors that affect the FLCs, including sheet thickness, test conditions, and strain path (Palaniswamy and Billur, 2012). It was observed that the

increase in the sheet thickness postpones the failure. This moves the FLC up along the vertical axis (major strain, ε_1). Also, radius of curvature of the hemispherical punch and the friction conditions influence the deformation of the material in the material test. Increase in the friction condition results in earlier failure of the sheet during the test, thus moving the curve down along the vertical axis. Therefore, test results from two different tooling dimensions in different laboratories may not be compared directly. Also, in the tests used to estimate the FLCs, the sheet material is subjected to a constant strain path until it fractures. However, in practical forming processes, the strain paths are not as constant as in the test. Therefore, the FLCs need to be used with much care in predicting failures in the forming process. It was observed that the FLC decreases (moves down in the vertical axis) when the sheet materials are initially subjected to positive minor strain ε_2 followed by negative minor strain ε_2 during deformation. FLC increases (moves up in the vertical axis) when the sheet materials are initially subjected to negative minor strain ε_2 followed by positive minor strain ε_2 during deformation (Palaniswamy and Billur, 2012).

The shape of the forming limit curve is not the same for all materials. The FLC based on the Keeler and Goodwin law can be used for most standard draw quality steels but may not hold for aluminum or special steel grades (Altair Engineering, 2011). Even for the same material, the FLC may vary due to the variation in the production process. An accurate FLC can be established by experiments that provide pairs of values of the limit strains ε_1 and ε_2 obtained for various loading patterns such as equibiaxial tension, biaxial tension, and uniaxial tension (Banabic, 2010).

A forming limit diagram generated in the forming simulation package Pam-Stamp (ESI Group, 2011) for a formed part is shown in Figure 13.12. Each point on the diagram is linked to the strain state of a specific finite element on the part. Note that Curve 2 is of the same shape as the FLC (Curve 1) but decreased 10% at $\varepsilon_2 = 0$. The region between Curve 1 and Curve 2 is referred to as the marginal zone in Pam-Stamp, indicating a risk of tearing; Curve 3 is a straight line given by

$$\varepsilon_2 = \frac{-(1+\overline{R})}{\overline{R}}\varepsilon_1 \tag{13.30}$$

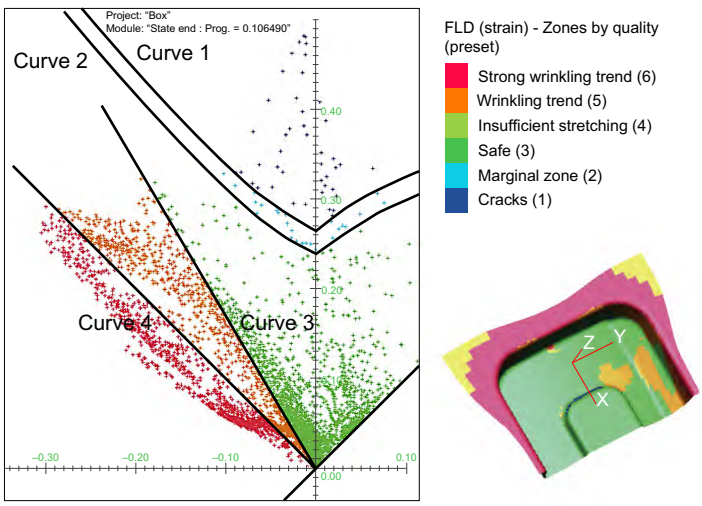


FIGURE 13.12

A forming limit diagram generated in Pam-Stamp (ESI Group, 2011).

where $\overline{R}$ is the coefficient of normal anisotropy. The elements between Curve 3 and Curve 4 (-45° line) will have the tendency to wrinkle. In general, the objective of sheet forming process design is to ensure that strains in the sheet stay in the safe zone and do not approach the limit curves.

13.2.5 SPRINGBACK ANALYSIS

In sheet metal forming, although permanent plastic deformation plays the major role, elastic strain always exists. When a formed part is removed from the die, the material has a tendency to partially return to its original shape on the release of the forming forces. This elastic shape change, or *springback*, is caused by the elastic recovery of the material and uneven stress distribution after forming.

The draw bending test (see Figure 13.13(a)), presented as a benchmark problem at the NUMISHEET 1993 conference (Zhou and Wagoner, 1993), is often used to assess springback in sheet metals under more realistic forming conditions. During forming, the blank material experiences stretching, bending, and unbending deformations when it passes the tool radius. This deformation path creates a complex stress state that is responsible for the formation of so-called sidewall curl (see Figure 13.13(b)). Various authors have used this experimental setup in their studies, and it has been shown that the sidewall curl becomes more pronounced for small tool radii and smaller clearances between the tools. As a result of springback, the geometry of the formed part will deviate from the shape imposed by the tooling, which may well give rise to problems for subsequent assembly operations.

In this subsection, we introduce the most fundamental mechanics behind the springback phenomenon in a simplified bending process. The objective is to provide readers with a basic understanding of springback using a simple bending example similar to that of Marciniak et al. (2002). A comprehensive introduction of this subject and detailed discussions on more complicated cases are presented in the work of Marciniak et al. (2002).

Consider a continuous sheet with unit width and thickness t as shown in Figure 13.14(a). The cylindrical bent region with curvature radius ρ and bend angle θ is formed by applying a pure bending moment M. When the sheet is wide enough compared to its thickness, which is usually the case,

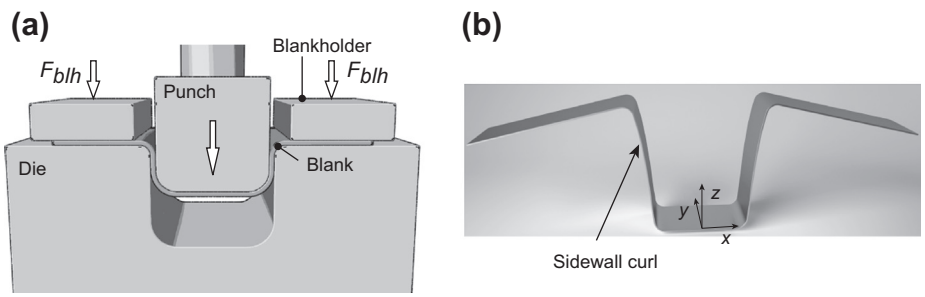


FIGURE 13.13

Top-hat section test - NUMISHEET'93 benchmark, (a) set-up schematic, and (b) blank after springback, mild steel.

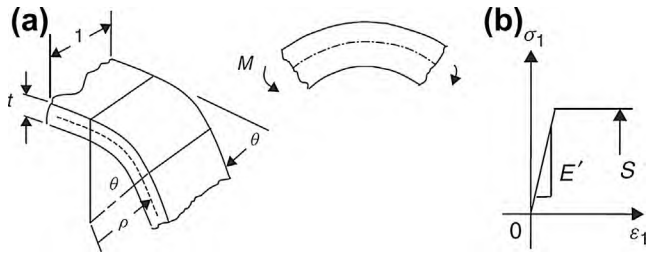


FIGURE 13.14

(a) A continuous sheet with unit width bent along a straight line, (b) elastic-perfectly plastic material model (Marciniak et al., 2002).

a plane strain bending condition can be assumed (i.e., the strain parallel to the bend is zero). For isotropic material, the plane strain bending condition is given by

$$\begin{aligned} &\text{Stress: } \sigma_1; \sigma_2 = \sigma_1/2; \sigma_3 = 0 \\ &\text{Strain: } \varepsilon_1; \varepsilon_2 = 0; \varepsilon_3 = -\varepsilon_1 \end{aligned} \tag{13.31}$$

In the meantime in this case, the material model is chosen to be *elastic-perfectly plastic* (Figure 13.14(b)), which is generally adequate for studying the process when the bend ratio (ρ/t) is greater than about 50 (Marciniak et al., 2002). The elastic modulus in plane strain bending can be determined with the uniaxial Young's modulus E and the Poisson's ratio v, as

$$E' = \frac{E}{1 - v^2} \tag{13.32}$$

Note that in this material model the plane strain flow stress S is a constant for any strain greater than the yield strain. If a von Mises yield condition is assumed, we have

$$\sigma_\theta = \frac{2}{\sqrt{3}}\sigma_f = S \tag{13.33}$$

where σ_f is the uniaxial flow stress and S is the plane strain flow stress.

Under the preceding given conditions, the stress distribution through thickness at the bend will be similar to that illustrated in Figure 13.15(a), provided that a linear strain distribution shown in

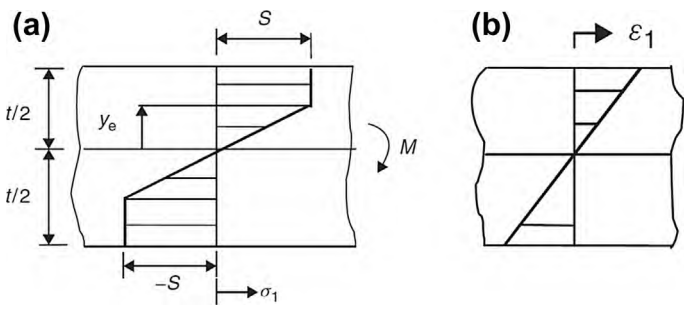


FIGURE 13.15

(a) Stress distribution for an elastic-perfectly plastic sheet in pure bending case, (b) linear strain distribution (Marciniak et al., 2002).

Figure 13.15b is assumed. If the material at $y = t/2$ exceeds the plane strain yield stress S, a critical distance

$$y_e = \frac{S}{E'}\frac{1}{(1/\rho)} \tag{13.34}$$

exists such that the material undergoes plastic deformation for $y > y_e$. Certainly, a curvature ρ_e can be found that gives $y_e = t/2$, which indicates a so-called elastic bending case.

Based on equilibrium, the moment required to bend the sheet, as shown in Figure 13.14(a), can be calculated, resulting in a moment–curvature curve as shown in Figure 13.16 (solid curve 0AB). Note that any point on *0A* represents an elastic bending. In addition, as the curvature grows, the curve on the right of point A approaches $M = St^2/4$.

Now we investigate the unloading process, as shown in Figure 13.17, for a sheet that has been bent to a particular curvature ρ_0 that is large enough so that the bending moment can be estimated as $St^2/4$. Note that ℓ is the constant length of the mid-surface at the bend. If this unloading process is assumed to be elastic, the unloading curve BC on the moment–curvature diagram will be parallel to the elastic loading curve, as shown in Figure 13.16. The proportional change in curvature obtained geometrically can be written as

$$\frac{\Delta(1/\rho)}{(1/\rho)_0} = -3\frac{S}{E'}\frac{\rho_0}{t} \tag{13.35}$$

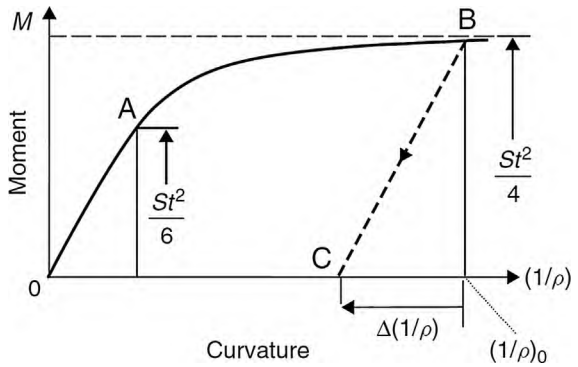


FIGURE 13.16

Moment–curvature curve (0AB) for an elastic-perfectly plastic sheet in pure bending case. Line BC represents a springback process (similar to that of Marciniak et al., 2002).

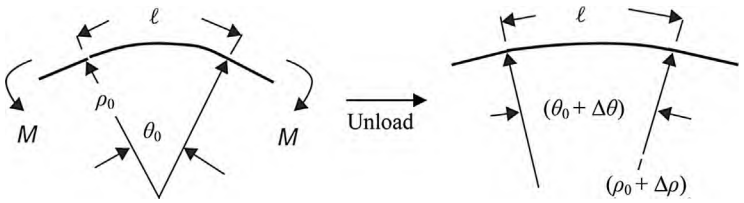


FIGURE 13.17

Unloading of a bent sheet (similar to that of Marciniak et al., 2002).

EXAMPLE 13.2

A 1 mm thickness wide metal sheet made of Aluminum 2024 has been bent to an angle of 30 degrees with curvature radius 25 mm. The elastic modulus, flow stress, and Poisson's ratio of the material are 73 GPa, 76 MPa, and 0.33, respectively. Assuming that the change in curvature will be small, calculate the deflection of the sheet due to springback in terms of change in bend angle.

Solution

The plane strain flow stress can be determined using Eq. 13.27

$$S = \frac{2}{\sqrt{3}} 76\,\text{MPa} = 87.8\,\text{MPa}$$

The plain strain elastic modulus can be obtained by applying Eq. 13.26

$$E' = \frac{73\,\text{GPa}}{1 - 0.33^2} = 81.9\ \text{GPa}$$

According to Figure 13.16, the relationship between the curvature and the bend angle is

$$\theta = \ell \frac{1}{\rho} \tag{13.36}$$

If the change in curvature or bend angle is small, we differentiate Eq. 13.36 to obtain

$$\frac{\Delta\theta}{\theta} = \frac{\Delta(1/\rho)}{1/\rho} \tag{13.37}$$

Substituting Eq. 13.35 into Eq. 13.27 yields

$$\Delta\theta = -3\frac{S}{E'}\frac{\rho_0}{t}\theta \tag{13.38}$$

Then the bend angle change of the sheet after springback can be approximated as

$$\Delta\theta = -3 \times \frac{87.8\ \text{MPa}}{81.9\ \text{GPa}} \times \frac{25\ \text{mm}}{1\ \text{mm}} \times 30^\circ = -2.4^\circ$$

From Eq. 13.35, it can be seen that the springback is proportional to the bend ratio (ρ/t), bend angle (θ), and the ratio of yield stress to elastic modulus (S/E'). In fact, it has been found that the springback deviations for HSS and aluminum are in general more severe due to their higher yield stress to modulus ratio (Banabic, 2010).

13.2.6 NUMERICAL IMPLEMENTATIONS

Numerical modeling of metal forming processes has now gained the industrial stage, and it has become possible to simulate metal deformation and to calculate stress and strain states for complex processes. By using the FEA method, simulation of design can help to predict errors and modifications can be made at an early stage before the tooling is fabricated and tested. Subsequently the labor cost and time lost can be reduced. Therefore, finite element methods will gradually replace manual trial-and-error design iteration with sophisticated numerical simulations.

The computer simulation of the forming process is conducted in two major steps (Firat, 2007). First, a forming analysis is conducted, including the blank and tooling, to determine the sheet metal

deformation during the forming process. Second, the sheet metal springback deformations following the removal of the tooling are computed using the forming stress distribution and the deformed geometry along with thickness distribution. There are some fundamental differences in the characteristics of both computation phases.

In the forming analysis phase, an initially flat sheet is placed between the tooling elements, usually involving the die, punch, and blank holder. The process is comprised of the complex interaction between blank (thickness and shape), the forming process (tooling, forming machine, forces, lubrication, etc.), and the material (ductility, material parameters, microstructure, resistance, residual stresses, etc.). The reason why the interactions in sheet metal forming processes are so complex is that a change in one area creates changes in the other areas and those interactions are highly nonlinear. Initially, the sheet is discretized into uniformly regular finite element mesh. Applying incremental displacement throughout the iteration steps was carried out along with the forming process. Subsequently, the parts of the mesh are refined to determine the contact points between the forming tools and the sheet metal in order to ensure smooth flows of material. Additionally, large plastic strains have to be modeled, using specially developed material laws, with corresponding stain hardening.

It is common, in sheet metal forming analysis, to include only the surface of the tooling in the FE model, rather than the complete solid geometry, as rigid geometric entities. In this modeling approach, the elastic deformation of the die elements is neglected, and this assumption holds for the majority of forming applications. As a result of the high aspect ratio with regard to the sheet blank geometry, shell elements are usually used in 3D FE models under the assumption of plane stress conditions throughout the forming analysis phase. The forming process forces are defined either in the form of a displacement-driven loading of rigid die surfaces or force-controlled loading of rigid blank holder surfaces.

In general, the forming process is controlled by the time-dependent interactions of the blank and tooling through a frictional contact-interface, and this results in gross shape changes of the sheet metal. Consequently, the computational modeling of the forming process necessitates an incremental formulation due to the geometrically nonlinear kinematics of sheet metal deformation involving large displacements, large rotations, and finite plastic strains. On the other hand, the springback deformations of a typical forming part are comparatively small, on the order of sheet thickness, and are mainly caused by the unbalanced through-thickness stresses of the sheet once it is taken out of forming tooling.

Two types of nonlinear analysis generally are included in finite element software: geometric nonlinearity that arises from significant changes in the structural configuration during loading and material nonlinearity in which effects arise from a nonlinear constitutive model (i.e., progressively disproportionate stresses and strains). Nonlinear geometric and material effects may be incorporated into this analysis. To achieve final equilibrium in a nonlinear analysis, the FE equations must be solved by constantly adjusting the applied forces based on the current state of equilibrium and modifying the geometry based on the current displacements.

With the progress of FE methods, along with the computational hardware and software technologies, the explicit and implicit incremental formulations have been developed for process modeling and analysis. The explicit dynamic and static incremental methods have found widespread use in the modeling and analysis of sheet metal forming because of its ability of better contact handling and relatively low computational cost when compared to the implicit static incremental method. The springback deformations of the sheet are computed using the deformed geometry of blank mesh along with forming stresses on the sheet as the basic inputs. Thus, the FE analysis of springback deformation

requires at least one forming step to be simulated. Both incremental explicit and implicit approaches can be used; however, the implicit static FE analysis is preferred since it ensures the convergence to a globally self-equilibrated stress state within the specified error tolerance.

The basic steps for conducting a representative FE simulation of a sheet metal forming process, including springback, are given schematically in Figure 13.18 and may be outlined as follows (Firat, 2004):

- Geometric modeling of tooling as rigid surfaces and geometric contact entities.
- Blank size estimation and geometric description.
- Constitutive modeling of the blank for elastic–plastic material response using the simple tension tests and forming limit curve of the material as inputs.
- Definition of frictional contact conditions between the blank and the tooling surfaces.
- Description of the forming process using the prescribed displacements or forces on the tooling surfaces.
- Incremental explicit static/dynamic or implicit static solution of the FE model.
- Postprocessing the time-history analysis results in terms of forming stress, forming loads, and formability diagrams, along with thickness distribution.
- Incremental implicit static or explicit dynamic (relaxation) analysis of the blank FE using the forming stress and deformed geometry along with the springback displacement boundary conditions.
- Postprocessing the (residual) production stresses and estimation of springback deformations.

13.3 PROCESS PLANNING AND TOOLING DESIGN

The bottleneck of the sheet metal forming development cycle is the tooling design lead time (Engineering Technology Associates, Inc., 2011). Due to the rapid development of FE modeling and CAE techniques during the past two decades, accurate and reliable sheet forming simulation programs are now available. It therefore becomes possible to transform traditional trial-and-error-based practice into a science-based and technology-driven engineering solution (Tisza et al., 2008). By integrating CAD and CAE systems, the simulation and validation of the full development cycle for sheet metal products can be realized.

This simulation-based process planning and tooling design process is illustrated with the flowchart in Figure 13.19. After the digital model of the product is created in a CAD system, the designer can perform a fast feasibility study with a one-step simulation procedure using FEA tools to determine whether the part can be formed without major defects. Once the part design is finalized, process planning and tooling design can be carried out, and then validated through incremental forming analysis. The tooling geometry and process parameters, such as binder force, may need to be modified or redesigned through several iterations based on feedback from simulations. Part deflection because of springback can also be predicted in simulations, and necessary corrections can be made to the tooling geometry to compensate for the springback effect. The manufacturing and tryout of toolings will be conducted after the simulation yields acceptable results that meet the design requirements.

With this approach, forming defects may be minimized and even eliminated before the real die construction stage. Improvement or redesign of the part and tooling can be completed within a very short turnaround time. As a result, the overall product development cycle, as well as the associated

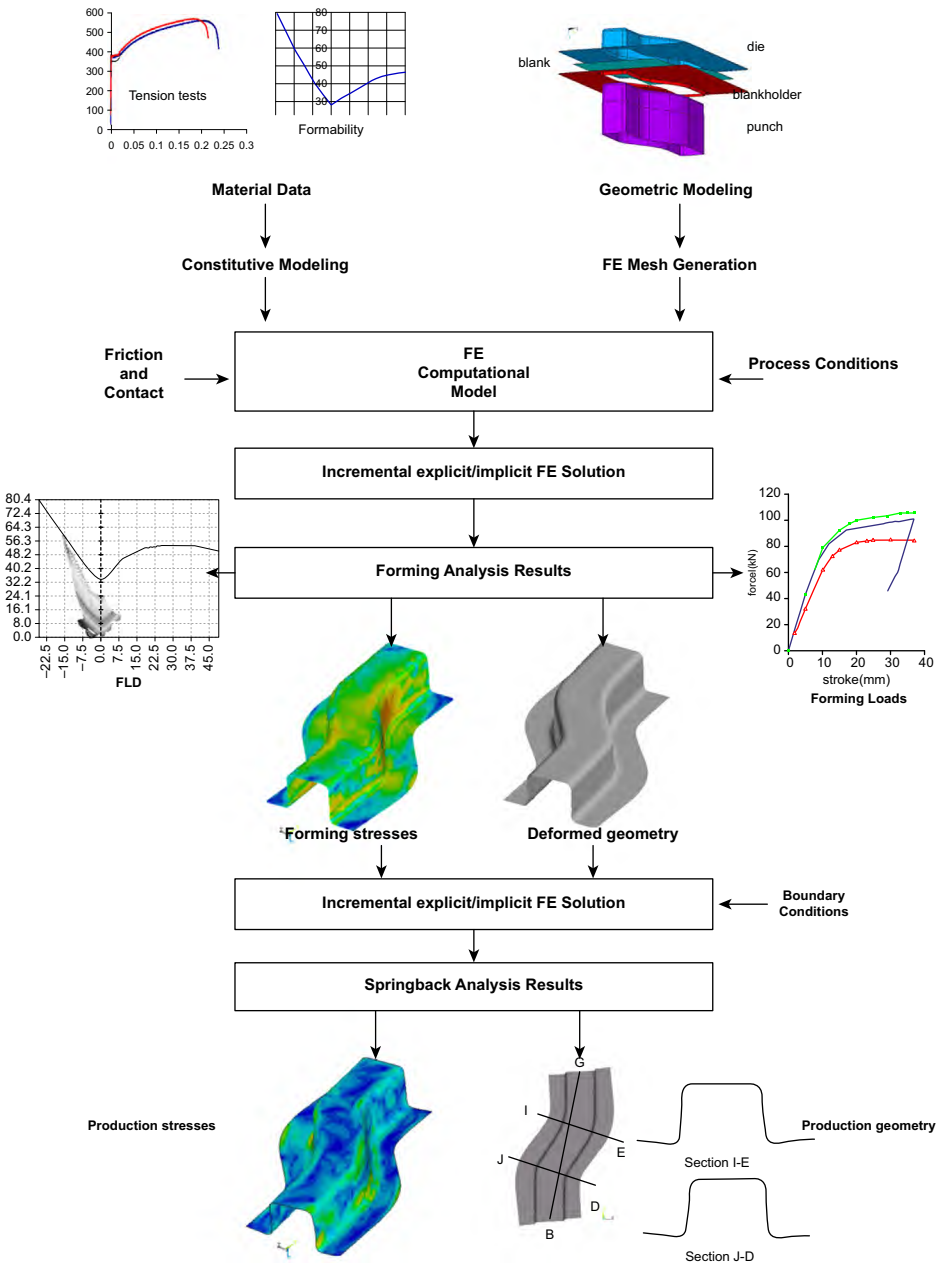


FIGURE 13.18

The FE simulation of sheet metal forming process including springback (Firat, 2007).

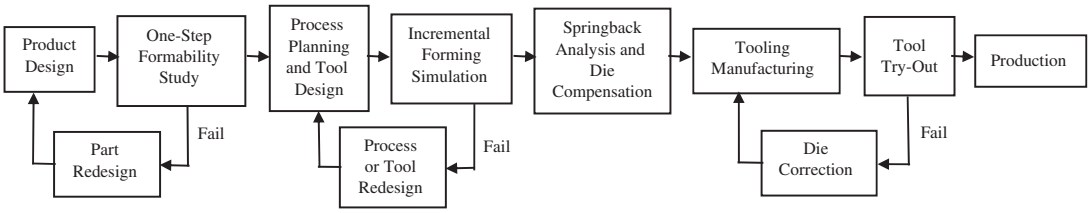


FIGURE 13.19

Flowchart of simulation-based process planning and tooling design process (Tisza et al., 2008).

costs, will be significantly reduced. In the meantime, high-quality sheet metal parts can be produced with optimized tooling designs and process parameters.

This section provides brief discussions on critical steps within the process shown in Figure 13.19. Formability study with one-step simulation will be introduced in Section 13.3.1. A typical tooling design procedure adopted in most forming software packages is given in Section 13.3.2. Section 13.3.3 briefly describes the incremental forming analysis techniques. Finally, springback analysis and springback compensation are discussed in Section 13.3.4.

13.3.1 ONE-STEP SIMULATION FOR FORMABILITY STUDY

A one-step simulation is primarily used to quickly assess part formability and estimate the blank outline in the early stage of the forming process design (Engineering Technology Associates, Inc., 2011). The starting point of a one-step simulation is the geometric model of the formed part. As soon as the part material and thickness are specified, the one-step solver (also called *inverse solver*) calculates the positions of the blank material points in the original undeformed flat plane. In other words, it "unfolds" the deformed blank to its initial undeformed configuration.

The characteristic feature of the one-step problem is that the unknowns are distributed between the initial blank and the formed part (ESI Group, 2011). For example, the known quantities in a one-step problem include the thickness, initial stresses and strains of the undeformed blank, and the geometry of the formed part. The unknowns are the locations of the blank material points on the flat blank surface in the initial configuration, as well as the thickness, stress, and strain distributions on the final part. A static, nonlinear problem can therefore be posed in mathematical terms.

The calculation of the problem is carried out without taking into account the shapes of the blank at any intermediate step in the forming process, and that is why it is called one-step (ESI Group, 2011). Figure 13.20 illustrates the initial blank shape of a sheet metal part solved by one-step analysis compared to its formed geometry. Once a displacement field is solved that ensures the equilibrium of the final configuration of the deformed blank, it is possible to compute the residual stress, strain, and thickness of the formed part. As shown in Figure 13.21, the forming limit diagram can also be generated, based on which a quick formability validation of the part can be conducted.

One-step solvers are designed to run very fast so that most simulations can be done within seconds. The simulation procedure provides valuable information about the feasibility of the design concept at the very first stages of product development. The blank shape can be calculated for estimating material cost. Formability of the part can be analyzed even without any tooling information. After running the

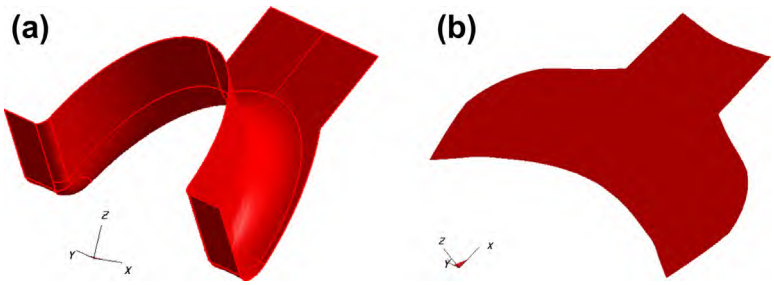


FIGURE 13.20

The 'unfolding' of a sample part, (a) final part shape, (b) initial blank shape.

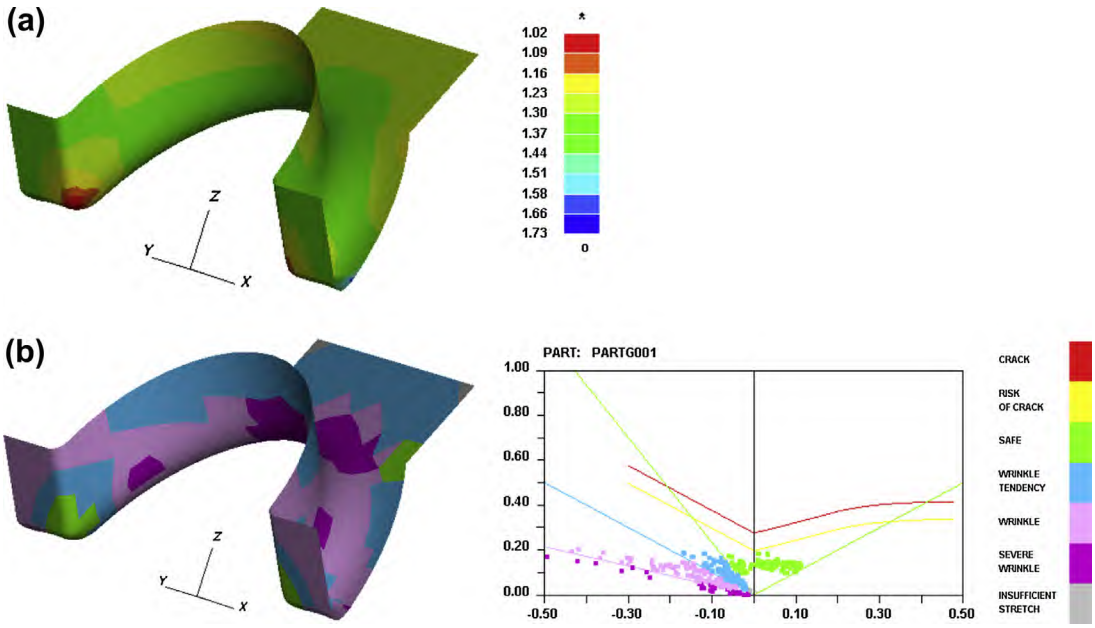


FIGURE 13.21

Results of one-step formability analysis for a sample part, (a) thickness distribution (mm) (initial thickness 1.3 mm), (b) forming limit diagram.

one-step simulation, a quick decision can be made as to whether any modification of part shape, material type, or blank thickness is required prior to die design.

However, it is important to know that one-step simulation results can only be considered as a first impression of component formability. Although it is possible to specify simple constraints or binder conditions during the analysis, the method is inherently a simplification of the physics of the actual forming process. In one-step analysis, the strain paths are assumed to be linear, since the blank

deformation at any intermediate step is not taken into account. In addition, the effect of the history of contact and friction between tooling and blank is neglected. Therefore, in certain cases, the accuracy of one-step simulation results can be quite low (Tekkaya, 2000). On the other hand, even if the one-step simulation results in good formability, the final decision on the whole process realization can be made only after performing a detailed incremental analysis based on actual tooling information and forming conditions (Tisza et al., 2008).

13.3.1.1 Blank Fitting and Blank Nesting

Once the shape of the blank is determined in a one-step analysis, blank fitting and blank nesting can be carried out to optimize the utilization of sheet metal material. These capabilities are integrated in most forming software packages.

The blank fitting function enables designers to fit the blank outline into a regular shape, in most cases a rectangle or trapezoid. The fitting process can be done manually or automatically. In the latter case, the rotation angle of the blank and the size of the rectangle or trapezoid are automatically determined by the program. Figure 13.22 shows the results of both rectangular fit and trapezoidal fit for the same sample part.

For parts that are designed for large-scale production, the arrangement of blank outlines on sheet metal coils is directly connected to material cost. Blank nesting is a tool that calculates the best nesting layout of the blank outline based on given constraints such as coil width. The designer can take material utilization as the optimization target to maximize the use of the metal sheet or coil.

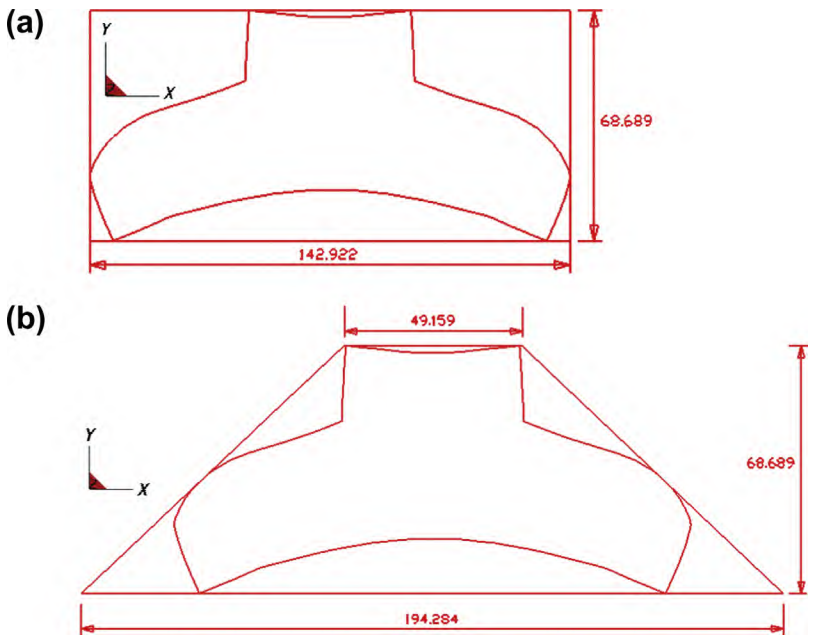


FIGURE 13.22

Blank fitting of the sample part shown in Figure 13.20(a), (a) rectangular fit, (b) trapezoidal fit.

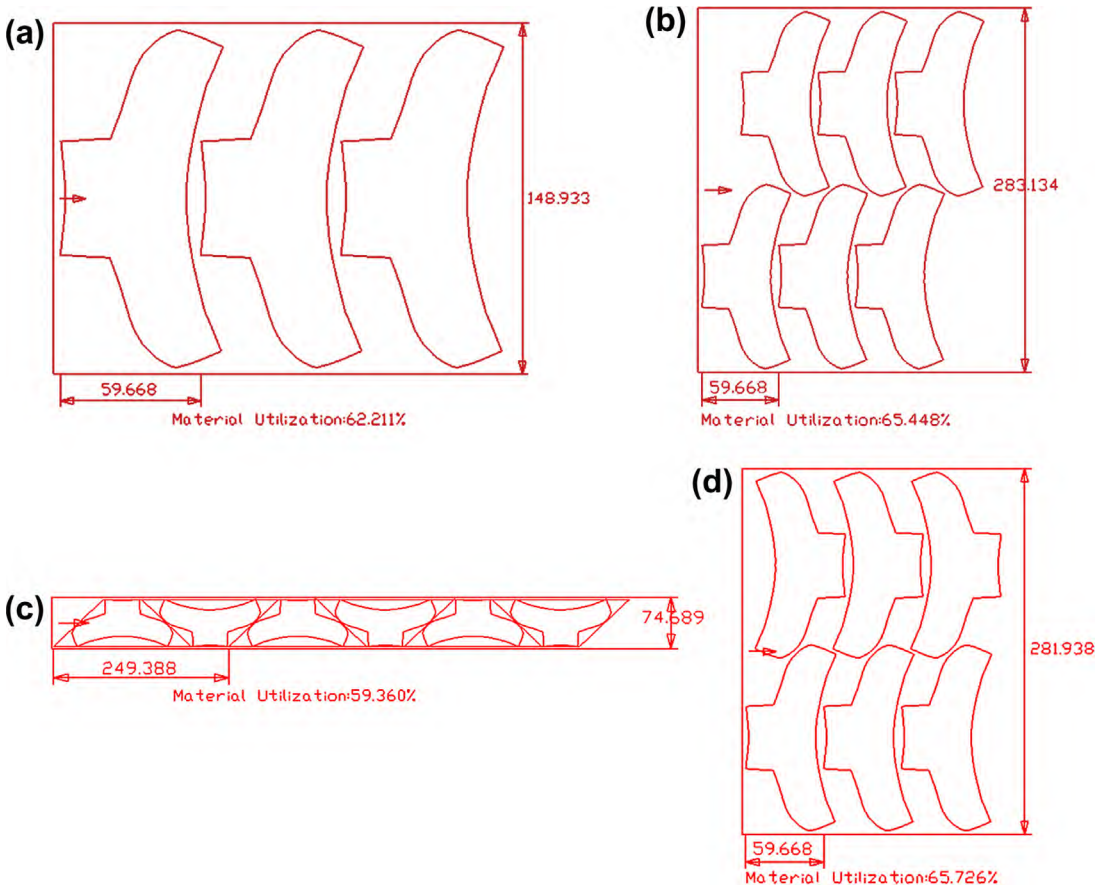


FIGURE 13.23

Blank nesting of a sample part, (a) one-up nesting, (b) two-up nesting, (c) isosceles trapezoid nesting, (d) two-pair nesting.

There are usually various nesting types available for designers to choose from. Some of them are illustrated in Figure 13.23. For example, the *Two-up Nesting* option (Figure 13.23(b)) generates the nesting result with two rows of blanks using a common orientation, while *Isosceles Trapezoid Nesting* (Figure 13.23(c)) arranges the blanks by embedding them in isosceles trapezoids (Engineering Technology Associates, Inc., 2011).

The blank fitting and blank nesting results can be exported in IGS format. The geometry model containing the blank outlines can be exported into CAD systems for further editing or directly imported into machining or cutting equipment (e.g., a water jet) for production.

13.3.2 DIE DESIGN

The main purpose of die design, or tooling design, is to generate accurate tooling models for manufacturing sheet metal parts. Long before the implementation of CAE in tooling design, CAD

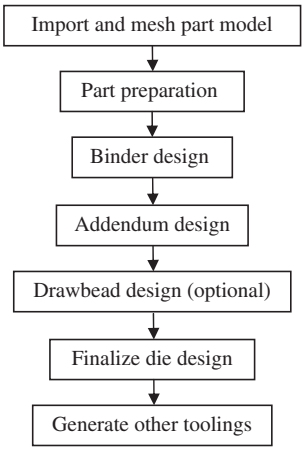


FIGURE 13.24

A typical tooling design process in using a forming software tool.

systems were adopted to create die geometries, which often result in poor die design and cause many unforeseen problems during tooling tryout (Tang et al., 2005a). On the contrary, forming software tools available today offer powerful functions tailored to early-stage die design for sheet metal products. Addendum surfaces and drawbeads can be automatically constructed based on key parameters and sketches. The final die face model created can be fully parameterized, enabling quick modifications to the design according to feedback from forming analysis. This simulation-based design process greatly reduces the iteration time for tooling design and development cycles as well as the associated cost.

The methodology of tooling design in most forming software packages is straightforward and intuitive. The die face construction process is usually divided into several steps, as illustrated in the flowchart of Figure 13.24.

13.3.2.1 Part Preparation

Before creating the tooling surfaces, it is important to ensure that the part geometry is well defined. Any area on the part that is not suitable to be taken into account as a die face should be edited or removed first. Usually, forming software tools provide integrated functions that allow designers to prepare necessary data for the part. Examples of such functions include unfolding flanges, removing holes, smoothing the part boundary, filleting sharp edges, and so on. If the half model of a symmetric part is imported, the part geometry needs to be mirrored. Some of the previously mentioned operations are illustrated in Figure 13.25.

In most forming simulations, the global Z direction is defined as the stamping direction. Therefore, a tipping operation is often necessary to adjust the part's orientation to follow the stamping direction. Besides manual adjustment, autotipping capability is available in most forming software to help designers orient the part based on a prescribed criterion such as minimizing draw-depth. In the meantime, it is essential to make sure that no undercut (back draft wall angle on die surface) exists before finalizing the part position.

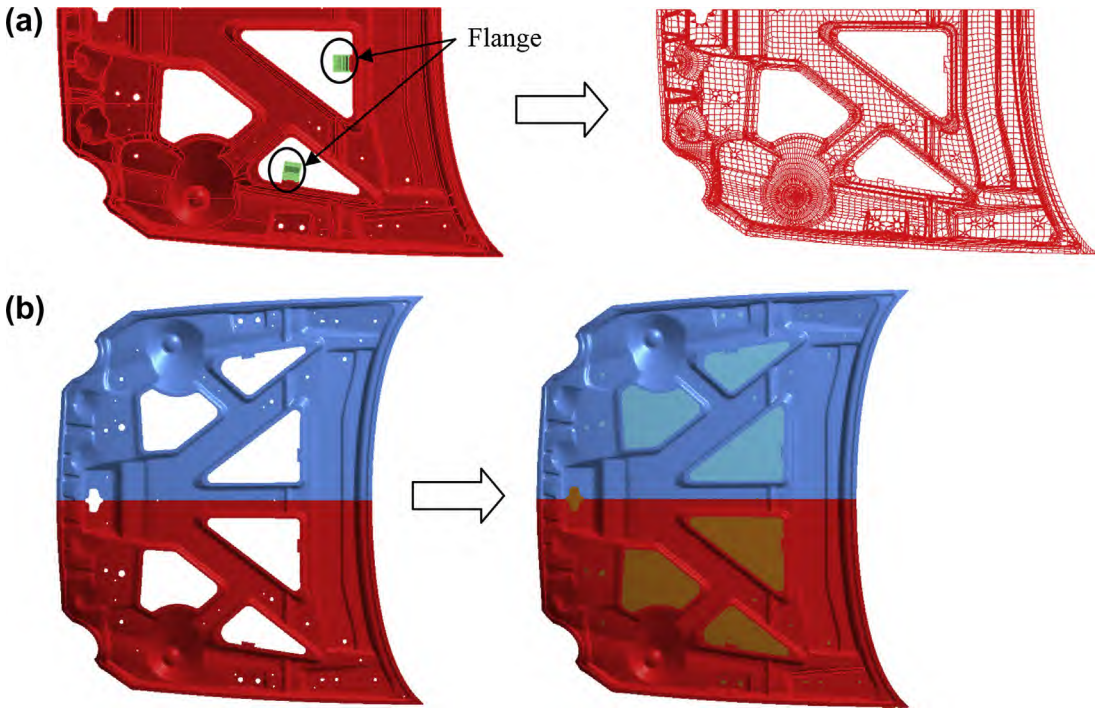


FIGURE 13.25

Examples of part preparation operations for a sample part, (a) removing flanges, (b) mirroring part and filling inner holes (Engineering Technology Associates, Inc. Tutorial).

13.3.2.2 Binder Design

Once part preparation is complete, a binder surface (or blank holder surface) needs to be defined. The binder surface is an important portion of the die face that controls the flow of material during a drawing process. The distance between the part and the binder surface determines the depth of the die cavity. Depending on the geometry of the part, the binder surface can be either a flat or a freeform curved surface based on the experience of the designer. Figure 13.26 shows a flat binder surface defined for the sample part shown in Figure 13.20(a).

13.3.2.3 Addendum Design

The creation of the addendum—that is, the portion of the die connecting the part area and the binder surface—is one of the most crucial steps in die design. The addendum surface is responsible for facilitating a smooth and controlled flow of the metal into the die cavity. With the help of forming software tools, the addendum can be generated automatically as a parametric surface based on a series of addendum profiles (or ribs) around the part outline, as shown in Figure 13.27(a). A profile is defined as the cross-section curve of the addendum that is usually cut along the outer boundary of the part at a direction normal to the boundary. The profiles can be adjusted separately to achieve a smooth and

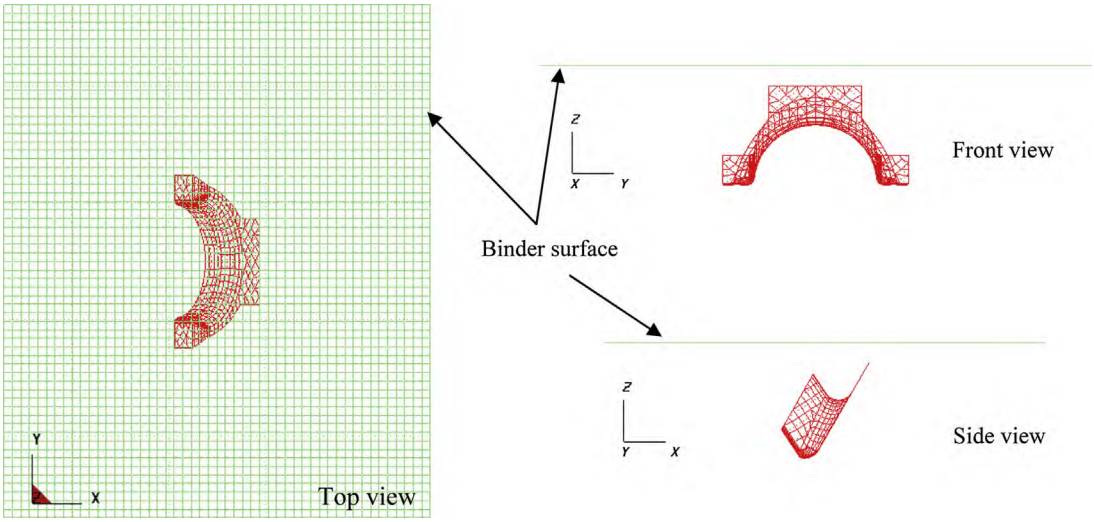


FIGURE 13.26

Three standard views of a flat binder surface designed for the sample part shown in Figure 13.20(a).

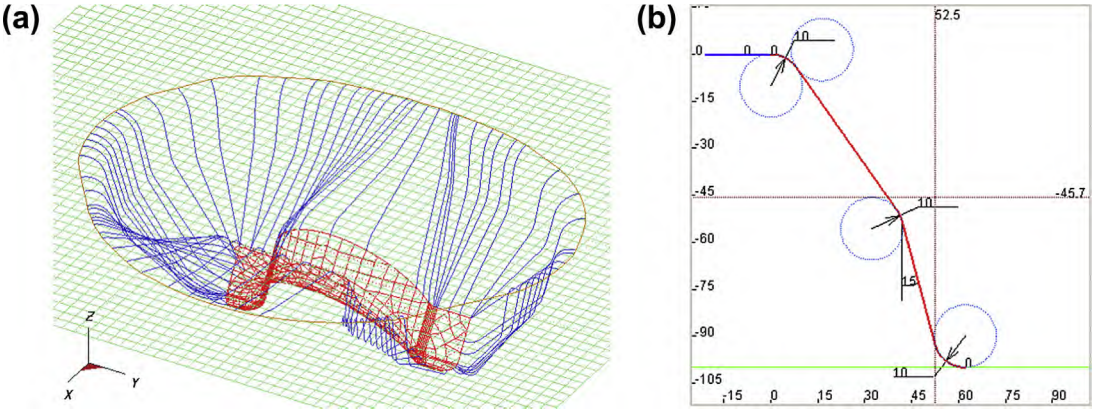


FIGURE 13.27

Addendum design for the sample part shown in Figure 13.20(a), (a) addendum profiles, (b) a parametric profile.

desired addendum surface. A parametric section profile with two segments is illustrated in Figure 13.27(b). By editing the shape parameters, key characteristics of the die face, such as wall angle and radius at transition areas, can be controlled precisely with maximum flexibility.

The addendum automatically built by software tools provides a rapid first shot for the addendum design and requires, in most cases, subsequent polishing such as smoothing the die opening line.

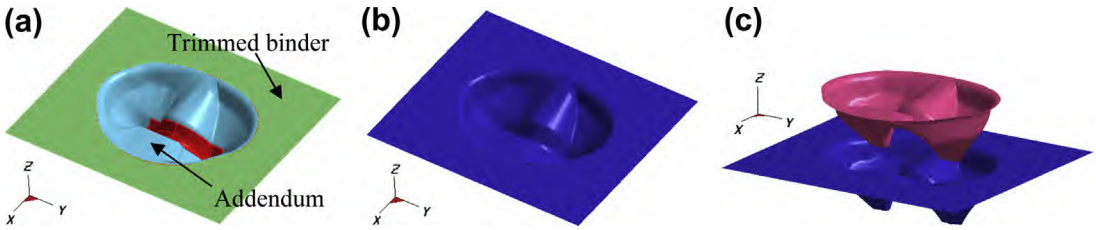


FIGURE 13.28

Die design for the sample part shown in Figure 13.20(a), (a) binder trim, (b) complete die face, (c) punch created based on die surfaces.

As soon as the addendum design is finished, the binder needs to be trimmed (Figure 13.28(a)) to get the complete die cavity (Figure 13.28(b)), which is the combination of the part, addendum, and trimmed binder surfaces. The geometry of the other part of the tooling can be obtained using associated portions of the die surface. For example, as shown in Figure 13.28(c), the punch is created by copying and offsetting the surface of the die cavity.

13.3.2.4 Drawbead Design

During a forming process, when a high restraining force is required to control the material flow, a large binder force must be applied, which may cause wear in the tooling. A control mechanism is therefore necessary to restrain local material flow sufficiently at a relatively low binder pressure. This can be achieved by applying drawbeads. The drawbead creates a restraining force by cyclically bending and unbending the sheet as it traverses the drawbead, causing strain hardening and a change in the strain distribution with consequential thinning of the blank (Altair Engineering, 2011). Generally, a pair of drawbeads consists of a male and a female portion located on different parts of the tooling (e.g., punch and die or binder and die), while no restraining force is applied on the blank until the die and punch are close to each other.

There are two common ways of creating drawbeads: *line bead* and *geometry bead*. A line bead is an equivalent drawbead model that represents the drawbeads analytically with line or curve segments. The drawbead restraining force is calculated based on the dimensions of the lines as well as the cross-sectional profile shape. The line bead is an efficient approach adopted in most forming simulation software, since modeling the exact drawbead geometry requires a large number of elements, thus increasing computation time drastically. The alternative approach is to create the drawbead mesh and surface according to the defined line bead and section shape so that the geometry of the drawbeads becomes part of the tooling surface.

The restraining forces calculated for geometry drawbeads are more realistic and accurate in general, since the interaction between the blank and the drawbeads will be simulated during forming analysis when the blank actually deforms as it passes through the drawbeads. The line bead and geometry bead can be easily converted to each other.

Figure 13.29 shows the geometry drawbeads generated based on a curve for a double-action draw process. The cross-section sketch of the drawbeads is given in Figure 13.29(b), with a male portion and a female portion located on the die and the binder, respectively. As can be seen from Figure 13.30, by applying drawbead structures, wrinkle can be reduced significantly without increasing the binder force.

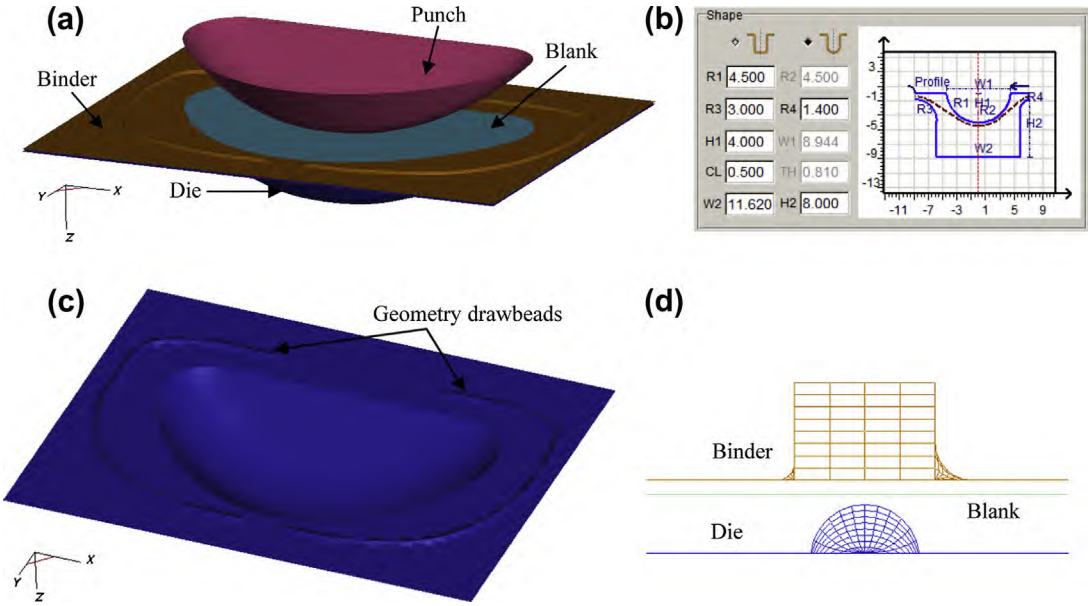


FIGURE 13.29

Curved geometry drawbead designed for a double-action draw process, (a) simulation setup, (b) cross-section sketch used to create the drawbeads, (c) drawbead structures (male portion) on the die face, (d) cross-sectional view of drawbead mesh.

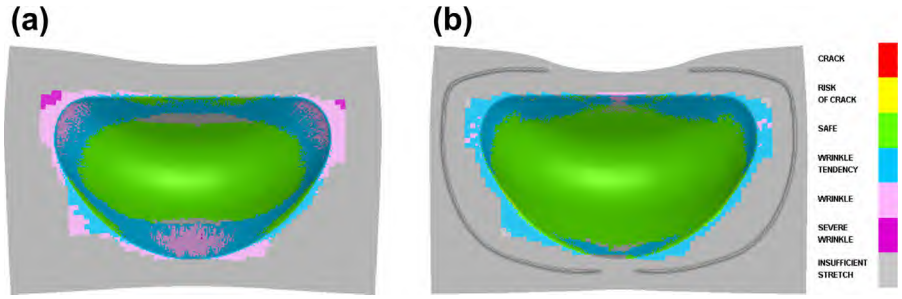


FIGURE 13.30

Simulation results before and after the drawbeads are applied (a) without drawbeads, (b) with drawbeads, most of the wrinkling has been eliminated.

13.3.3 INCREMENTAL FORMING ANALYSIS

Unlike one-step simulation, in which only the final part shape is solved, incremental forming simulation allows the accurate modeling of sheet forming processes. In incremental simulations, the forming process is divided into steps or increments, and the problem is solved incrementally through

time. Within each step or increment, the positions, velocities, accelerations, and forces are calculated at the nodal points in the blank. Within each blank element, stresses and strains can be evaluated from nodal displacements. As a more rigorous modeling approach, the incremental analysis allows designers to detect the stage at which process defects occur in the blank during the forming operation.

In general, two types of methods—*implicit* and *explicit*—can be applied for solving incremental forming simulations. The implicit method (or static implicit method) was the very first one used in simulation of sheet forming processes; using it the dynamic body forces are neglected and the problem is assumed to be quasi-static (Tekkaya, 2000). In the implicit method, the displacement and load conditions are broken up into several increments. For each increment, a set of nonlinear static equilibrium equations are solved with standard numerical methods such as Newton iteration (ESI Group, 2011). The implicit method enables a full static solution of the deformation problem with convergence control; however, the amount of calculation in each increment is large. In addition, memory requirements are high due to the matrix inversion step and accurate integration schemes. The implicit method also suffers the problem of divergence of the solution because of continuous change of contact and friction states, as well as the problem of singularity of the stiffness matrix at bifurcation points, such as instabilities at wrinkling initiation (Makinouchi, 1996; Tekkaya, 2000).

The explicit method (or dynamic explicit method) is based on dynamic equilibrium equations and makes use of time steps instead of increments to discretize the calculation. In this method, the stiffness matrix is not necessarily to be constructed and solved, so the computation speed is higher and the memory requirement is lower than with the implicit method (Makinouchi, 1996). Since the most standard time-step size for the explicit method takes around 10^{-6} seconds to satisfy the stability condition, most explicit simulations are performed under accelerated tooling speed to reduce computation time; this may not work if the material is strain rate sensitive. Alternatively, the density of the material can be artificially increased so that larger time-step sizes can be applied. One disadvantage of the explicit method is the lumped mass matrix and single-quadrature elements used, which may deliver relatively poor stress and strain accuracy (Tekkaya, 2000). Moreover, the absence of convergence control is a critical issue. Finally, it is noteworthy that springback simulations are usually solved using implicit algorithms due to the high nonlinearity of the problem.

The incremental analysis requires a complete set of inputs to ensure a precise modeling of the forming processes. For instance, the blank shape and all tooling surfaces need to be imported; material properties, including stress–strain curves, parameters for the given yield criteria, forming limit diagram curves, and so on, need to be accurately defined; tooling motions and tooling loads are also necessary to control the process. In addition, an accurate description of friction at the interface between the blank and the tooling is important for calculating the material flow and the forming loads. In most forming software packages, Coulomb friction is used to model realistic interface conditions (Tisza, 2004).

Once the incremental analysis is complete, based on detailed information on the tooling and process parameters, the resulting data at any intermediate step of the forming operation can be displayed for formability validation. Figure 13.31 illustrates the incremental simulation results for a sample part, including the stress and thickness distributions, as well as the forming limit diagram. Moreover, the deformation of the blank during forming can be recaptured through real-time animations, (e.g., see Figure 13.32).

Besides traditional single- and double-action drawing processes, sheet forming simulation software today supports many other types of forming applications such as sheet and tube

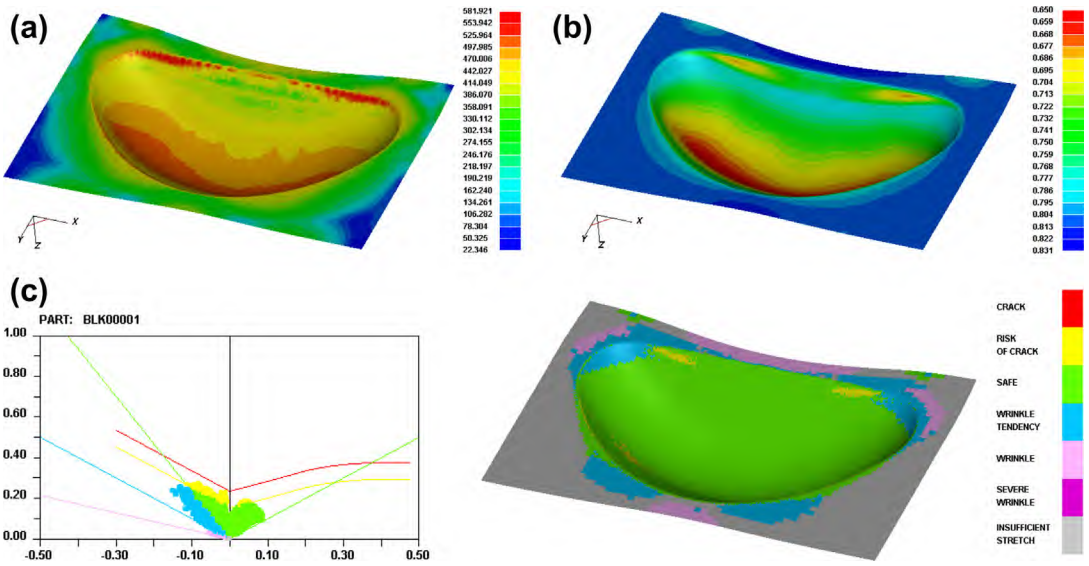


FIGURE 13.31

Incremental forming simulation results for a sample part at the final stage, (a) von-Mises stress distribution (MPa), (b) thickness distribution (mm), (c) forming limit diagram.

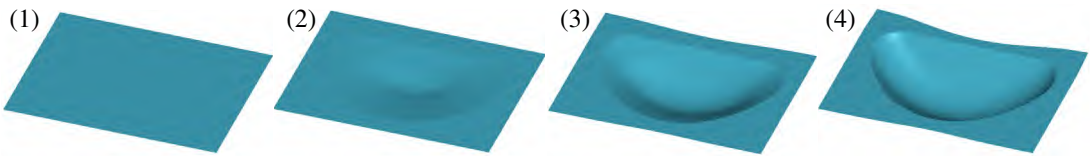


FIGURE 13.32

Blank deformation at different intermediate forming stages.

hydroforming, stretch forming, tube bending, roller hemming, and rubber forming. They also offer multistage analysis capabilities so that several forming operations can be simulated sequentially without user interactions. During forming simulation, mesh refinement can be adopted to locally adjust blank mesh density in order to capture details of tooling geometry. A high refinement level generally contributes to better accuracy in calculating the deformation and stress on the blank at localized areas; however, it also leads to a reduction of the step size and therefore longer simulation time.

13.3.4 SPRINGBACK ANALYSIS AND DIE COMPENSATION

In sheet metal forming, although permanent plastic deformation plays the major role, elastic strain always exists. When a formed part is removed from the die, the material has a tendency to partially

return to its original shape on the release of the forming forces. This elastic shape change, or *springback*, is caused by the elastic recovery of the material and uneven stress distribution after forming.

The deformation due to springback often results in components that do not fit in the assembly, while residual stress will be introduced into the assembly if sprungback parts are forcibly installed. Traditionally, springback can be resolved through manual adjustment on the shop floor. However, these processes lead to additional cost and assembly time.

Many of the forming simulation software available today are capable of predicting the deflection of the formed part due to springback. In addition, they offer the capability to modify the tooling surfaces based on springback analysis results to ensure that the part after springback meets design requirements. This capability, usually referred to as *springback compensation* or *die compensation*, is available in major software packages such as AutoForm (www.autoform.com), HyperForm (www.altairhyperworks.com), and DynaForm (www.eta.com/dynaform).

The shape of the part and its stress–strain state at the end of a forming simulation are the inputs required for a springback analysis. If necessary, the blank can be trimmed prior to performing the springback simulation. Due to the fact that no tooling needs to be considered in a springback analysis, the part must be constrained in some way to eliminate rigid body motion. One approach commonly used in simulation software is to apply fixture constraints to three nodal points, as illustrated in Figure 13.33. For instance, the translation degrees of freedom in all three directions for node A are fixed. For nodes B and C, translation movements in *XZ* and *Z* are fixed, respectively. By constraining these six translation degrees on freedom, rigid displacement of the blank during springback analysis can be avoided, while the part is still allowed to deform as a result of the release of residual stress. Figure 13.34 shows the comparison between the desired shape of a sample part (shape at end of drawing) and the shape of the same part after springback; this was solved using DynaForm, where the displacement to the desired part shape is plotted, indicating a maximum springback deflection of approximately 42 mm.

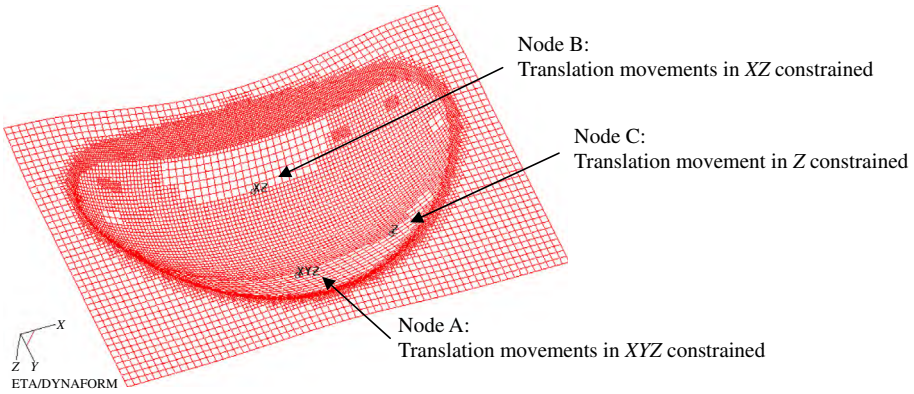


FIGURE 13.33

Nodal constraints required for springback analysis.

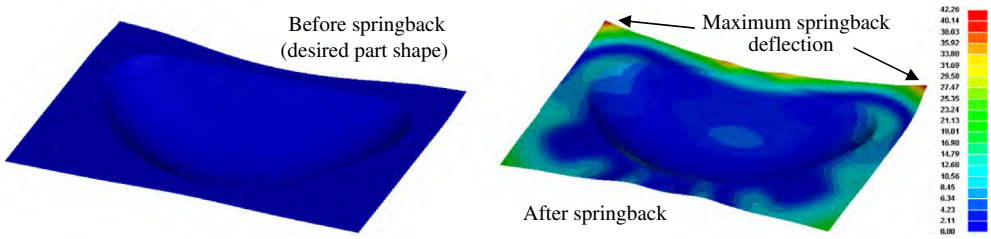


FIGURE 13.34

Shape of a sample part before and after springback.

Usually, springback analysis is defined as the last operation in a multistage incremental analysis. After the sprungback shape of the part is obtained, designers can fit the part geometries before and after springback so that the magnitude of the deflection can be observed through a deviation check.

The major purpose of springback analysis is to predict the deformation of the part, based on which toolings can be compensated to reduce the deviation of the sprungback part from its desired geometry. The essential idea of die compensation is to modify the original die face in a direction opposite to the springback to ensure that part dimensions fall within the tolerance of the intended part design after springback.

For decades, the compensation of tooling depended on the experience of engineers. With advanced development in simulation technology, springback compensation can now be quickly performed with the help of forming software tools. In springback compensation, the designers need to provide the geometry of the deformed part before and after springback, as well as the original die face. The original tooling geometry will then be morphed by applying a displacement field in a direction opposite to the springback deformation, with a scaling factor defined based on users' experience. Special care is often taken in software algorithms to avoid creating undercuts or negative drafts as a result of the compensation (Altair Engineering, 2011). If the desired blank shape cannot be obtained after the first compensation, several iterations may be performed to obtain acceptable tooling surfaces. A typical simulation–springback–compensation procedure is illustrated in Figure 13.35.

The geometry of the punch and the binder will be compensated simultaneously in a springback compensation calculation. The compensated tooling geometries may need to be repaired manually using CAD tools to remove local defects and improve the quality of surfaces. Afterwards, the compensated tooling can be exported into CAD systems for the tooling manufacturing processes that follow.

Springback compensation has been proved effective to alleviate excessive springback phenomenon in forming operations. Figure 13.36 illustrates a benchmark simple U-Channel example reported in Tang et al. (2005b), where only half of the model is shown as a result of symmetry. As we can see, the part after forming matched the original die face; however, it sprung backward by approximately 9 degrees after elastic stress recovery. The forming and springback analysis was then conducted using the tooling after the first compensation with a relatively large scaling factor, which resulted in a slight spring forward of the part. With a reduced scaling factor, the second springback compensation iteration was performed, and the formed and sprungback part with the new tooling became close to the desired shape.

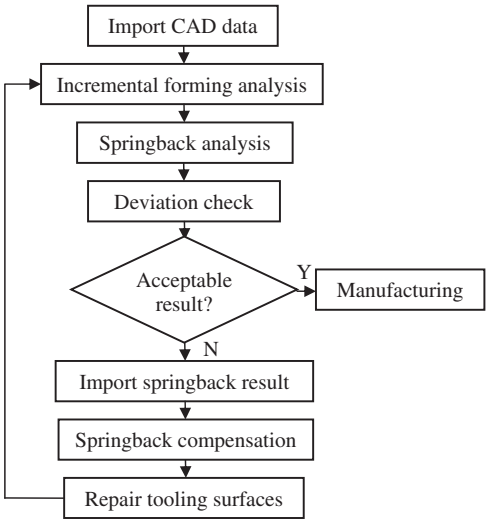


FIGURE 13.35

Flowchart of simulation–springback–compensation procedure.

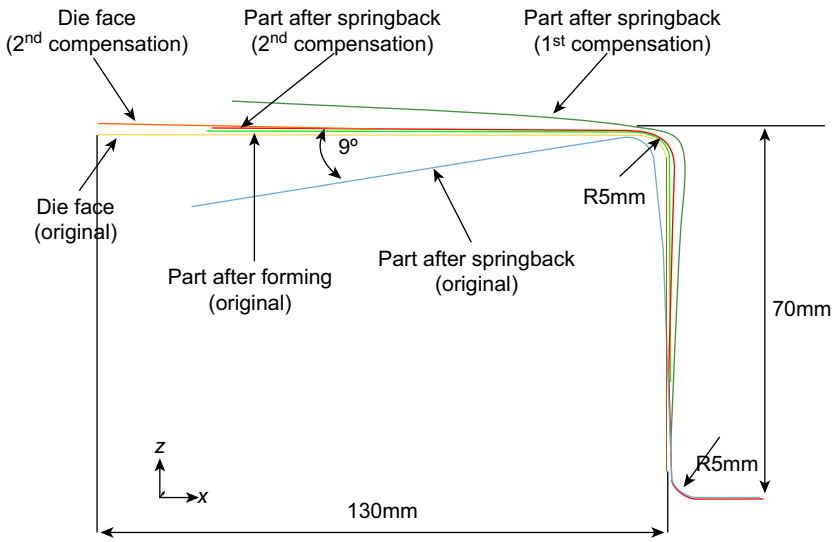


FIGURE 13.36

Springback compensation of a U-channel (Tang et al., 2005b).

13.4 COMMERCIAL FORMING SIMULATION SOFTWARE

In this section, we discuss commercial forming simulation software. In Section 13.4.1, we provide a short overview with more discussion on three codes—FastForm, AutoForm, and Pam-Stamp. They offer a good suite of forming simulation and tooling design capabilities and are considered leading software in support of sheet metal forming simulation. We then discuss, with more insight, two widely employed codes in the automotive industry, HyperForm and DynaForm, in Sections 13.4.2 and 13.4.3, respectively.

13.4.1 OVERVIEW OF SIMULATION SOFTWARE

Several commercial forming software tools are being used by industry. As listed in Table 13.1, these software tools include FastForm (www.forming.com), DynaForm (www.eta.com/dynaform), ABAQUS (www.3ds.com/products/simulia/portfolio/abaqus), LS-Dyna (www.lstc.com/products/ls-dyna), AutoForm (www.autoform.com), Pam-Stamp 2G (www.esi-group.com/products/metal-forming/pam-stamp-2g), QForm 3D (www.qform3d.com), Deform (www.deform.com), HyperForm (www.altairhyperworks.com/Product,4,HyperForm.aspx), OpenForm (gns-mbh.com/openform.html), and Stampack (www.quantech.es/stampack.aspx).

ABAQUS and LS-Dyna are general-purpose and highly respected FEA codes. These two codes serve well for the general FEA community, however, they are probably too broad to adequately support metal forming simulations. They do not offer important capabilities (Table 13.1), such as blank design and die design; in addition they have a relatively steeper learning curve. QForm 3D and Deform are inherently developed for simulating forming operations for metal blocks such as forging, rolling, and extrusion. OpenForm is more like pre- and post-processors that are decoupled from particular finite element codes for forming simulation (although it also has its own FE solver named "Indeed").

Table 13.1 Commercially Available Forming Simulation Software Tools

	Multistage forming process	Formability analysis	Blank design	Die design	Springback analysis
FastForm	√	√	√	×	√
DynaForm	√	√	√	√	√
ABAQUS	√	√	×	×	√
LS-Dyna	√	√	×	×	√
AutoForm	√	√	√	√	√
Pam-Stamp 2G	√	√	√	√	√
QForm 3D	√	√	×	×	×
Deform	×	×	×	×	√
HyperForm	√	√	√	√	√
OpenForm	√	√	√	×	√
Stampack	√	√	√	×	√

OpenForm supports major forming simulation software such as AutoForm, LS-Dyna, and Pam-Stamp. However, OpenForm does not have a die design capability, which is the same drawback as Stampack. It is worth noting that many forming simulation codes (e.g., DynaForm and HyperForm) employ LS-Dyna to formulate and solve FE equations for support of forming simulations.

Three codes, FastForm, AutoForm, and Pam-Stamp, which provide excellent forming simulation and tooling design capabilities, are discussed next.

13.4.1.1 FastForm

Founded in 1989, Forming Technologies Inc. (FTI®) providers software solutions for the design, feasibility, and costing of sheet metal components through FastForm. FTI has provided OEMs and suppliers in the automotive, aerospace, electronics, and appliance industries with solutions designed to reduce development time and material costs. Software tools that FTI offers include a forming suite that supports blank design, multistage forming, incremental simulation, and blank cost estimates. The company also provides CAD connections for blank design, including CATIA V5-based, SolidWorks-based, and Pro/ENGINEER-based capabilities.

One of the design modules, FastForm Advanced, offers formability analysis, which considers component or tool geometry and accounts for material properties, friction, binder surface, die addendum, blank holder force, pad pressure, drawbeads, and tailor-welded blanks. In addition, the Process Planner offers a stamping knowledge-based system for process planning and die cost estimation. In general, FastForm offers excellent capabilities for support of forming simulation, die design, and process planning. One drawback is the lack of springback analysis and die compensation. Furthermore, it only provides limited materials and crude yielding criteria, and offers fewer forming operations.

13.4.1.2 AutoForm

AutoForm was founded in 1995 in Zurich, Switzerland. Since then, AutoForm has grown to be recognized as one of the leading providers of software solutions for die design and sheet metal forming simulation. AutoForm offers fairly complete software solutions for the die making and sheet metal forming industries. Software modules offered include forming simulation, process planning, die design, blank design and trim line determination, springback analysis and die compensation, and cost analysis. In addition, some capabilities (e.g., one-step formability analysis and forming simulation) are integrated with CATIA V5 and NX, and blank design capabilities tied with Autodesk® Inventor®. AutoForm supports a good range of forming processes. In addition to basic processes, such as draw forming, the software supports sheet hydroforming, hemming, and so on.

13.4.1.3 Pam-Stamp 2G

Pam-Stamp 2G is a software module for sheet metal forming developed by the ESI Group founded in 1973, a pioneer and one of world's leading providers in Virtual Prototyping. Pam-Stamp 2G provides an integrated, scalable, and streamlined stamping solution. It covers the entire tooling process from quotation and die design through formability and try-out validation, including springback prediction and correction.

Forming processes supported by Pam-Stamp 2G are fairly complete, including draw forming, hot forming, hydroforming, line die, progressive die, rapid die design, rubber pad forming, roll hemming, simulation for production validation, stretch forming, superplastic forming, springback compensation,

and tube forming. One unique capability offered by ESI is its CATIA V5 PLM-based metal forming portfolio that provides both metal forming part designers and die designers with a set of engineering solutions that address design and manufacturing issues directly within a generative modeling PLM environment. This capability consists of two modules: Pam-TFA (Transparent Formability Analysis) which offers part feasibility study and part costing solutions, and Pam-Diemaker, which is a dedicated Workbench presented inside the CATIA V5 environment.

Although Pam-Stamp 2G currently offers many powerful capabilities, it does not have blank fitting and blank nesting capabilities. It does not offer cost estimating on blank or tooling either.

13.4.2 HYPERFORM

Developed by Altair Engineering, HyperForm is a FE-based sheet metal forming simulation module in HyperWorks, which is an enterprise simulation solution for rapid design exploration and decision making. HyperWorks provides a tightly integrated suite of software modules for modeling, analysis, optimization, visualization, reporting, and performance data management. In addition to LS-Dyna, HyperForm offers its own FEA solver, RADIOSS™, which allows users to quickly predict wrinkles and splits prior to cutting steel, avoiding the unnecessary costs associated with die machining and press downtime.

As an integrated module in HyperWorks, HyperForm offers formability analysis, which provides users a quick check in formability of the part design by conducting a one-step analysis and reviewing a forming limit diagram (FLD). It also contains die design that includes addendum design and incremental simulation that accurately models important metal forming processes throughout the forming simulation. Plus, there is support for die compensation to minimize part springback, as illustrated in Figure 13.37.

In addition to these crucial capabilities, HyperForm offers excellent design capabilities, including drawbead design, optimization for one-step analysis, and die stress analysis and topology optimization. Optimization for one-step analysis allows users to move strain points closer to a safe zone by adjusting process variables such as binder force, drawbead force, and so forth. Die stress analysis supports users in transferring tool contact forces from stamping analysis to the die model and performs stress analysis, as illustrated in Figure 13.38(a). Topology optimization supports a lightweight tooling design by carrying out density-based topology optimization (Bendsøe and Sigmund, 2003) using stress analysis results. One such example is shown in Figure 13.38(b).

13.4.3 DYNAFORM

DynaForm, developed by Engineering Technology Associates, is a simulation software solution for sheet metal forming. It allows the user to do formability analysis and cost estimation for product manufacturing. DynaForm offers a fairly complete set of forming operations pertinent to practices in the automotive and aerospace industries, including blank size engineering to estimate blank size. In addition, it performs several important tasks, including blank nesting, die face engineering, formability analysis, and die system analysis. Blank nesting supports maximum material usage, scrap and piece pricing. Die face engineering supports tooling design, including binder and addendum, from the part geometry. Die system analysis analyzes scrap shedding/removal, die structural integrity and sheet metal transferring and handling.

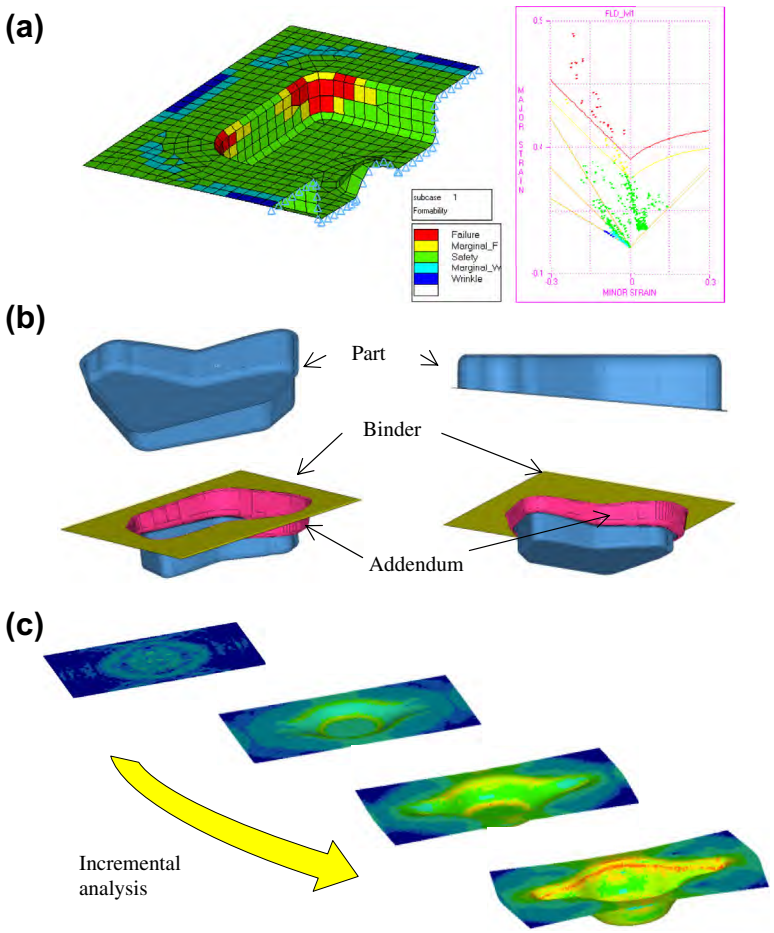


FIGURE 13.37

Major capabilities offered in HyperForm, (a) one-step analysis with forming limit diagram, (b) parametric addendum design for die, and (c) forming simulation: incremental analysis.

Both HyperForm and DynaForm perform well for regular draw forming. For example, the sample part shown in Figure 13.39 was simulated using both software modules with almost an identical setup, including coordinate systems, tool surface, blank size, blank orientation, and blank location with respect to the tools. Finite element meshes created for the tools are different in HyperForm and DynaForm. Furthermore, the underlying element formulations are different. In addition, the yielding criteria employed are Barlat and Hill for DynaForm and HyperForm, respectively.

As shown in the forming limit diagrams in Figure 13.39, both software tools predict similar results. Some tearing occurs around the drawbeads (in red) and small wrinkles (in light blue) occur around the four corners on the outside of the actual formed part.

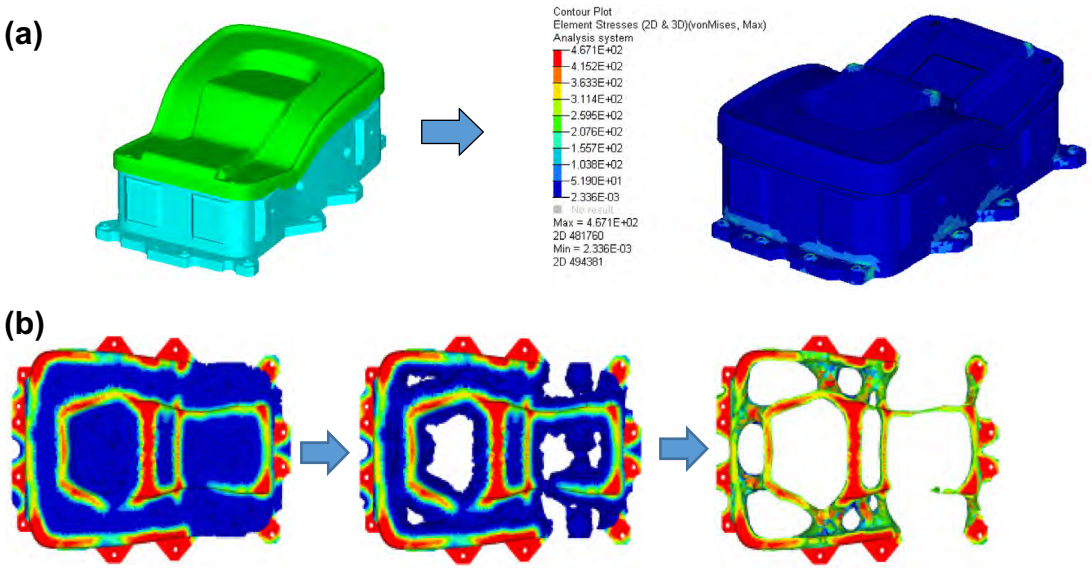


FIGURE 13.38

Design capabilities offered by HyperForm, (a) die stress analysis and, (b) die topology optimization.

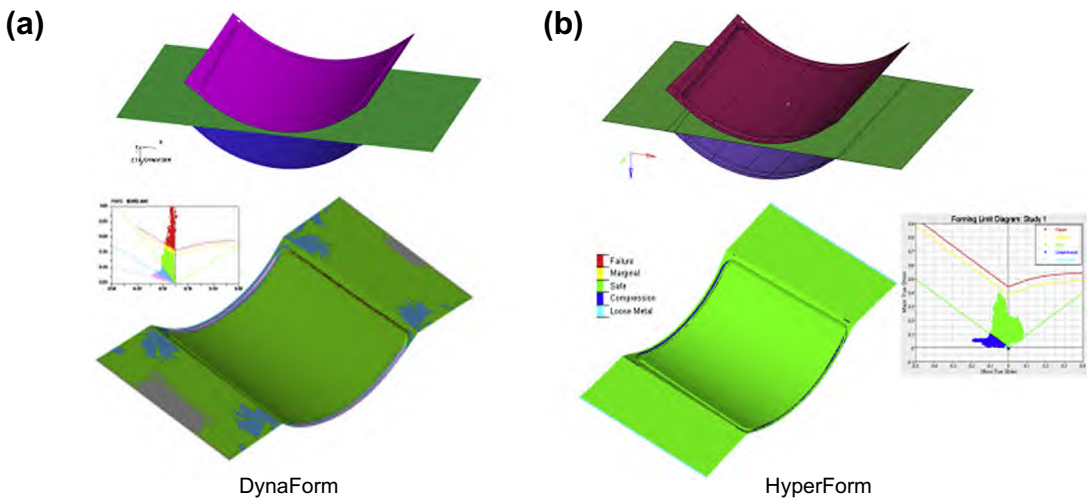


FIGURE 13.39

Forming simulations for sample part, (a) using DynaForm, and (b) using HyperForm.

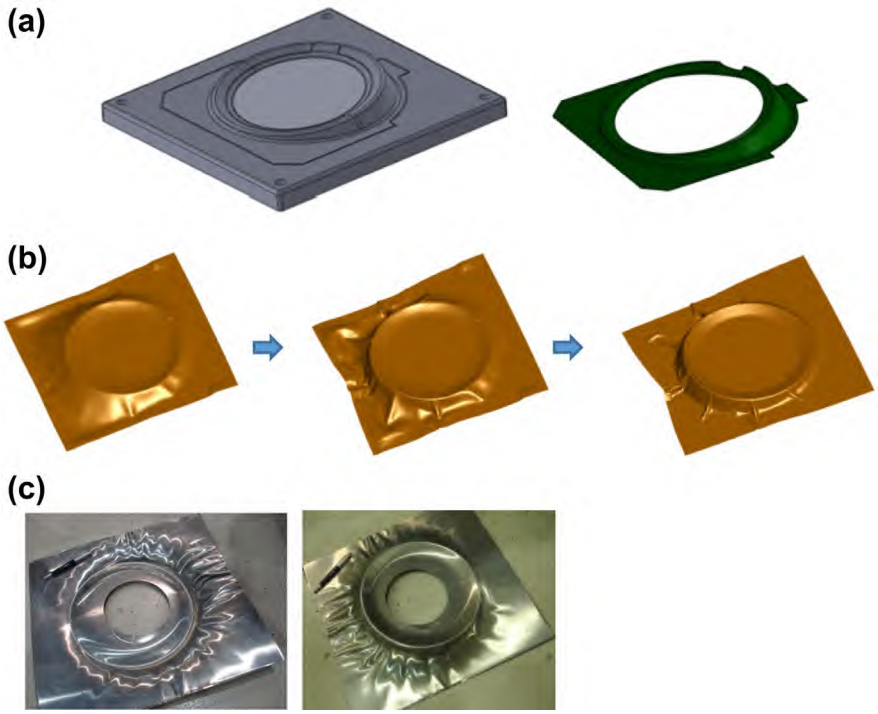


FIGURE 13.40

Sheet hydroforming for sample part, (a) die and part, (b) forming simulation, and (c) actual formed part (two views).

As for sheet hydroforming, DynaForm offers dedicated options for users to create and conduct forming simulations—for example, the sample parts shown in Figure 13.40. Note that the simulation accurately predicted severe wrinkles outside the circular ring, which is apparent in the actual part (Figure 13.40(c)) formed by using a fluid cell.

In addition, stretch forming is well supported in DynaForm. Using DynaForm, the jaw can grip the edges of the sheet and form curves that deform the blank. The punch can then move upward to deform the blank to a desired shape. If necessary, a female die can also be used to make the blank conform more closely to the desired part shape, as illustrated in Figure 13.41.

13.5 CASE STUDIES

Two case studies are included in this section: a core panel and a wheel fairing. In the core panel example, we focus on the technical aspects of using forming simulation to bring more knowledge to help engineers understand the physics and mechanics of the metal forming process. The second example offers more insight into using simulation software, in this case DynaForm, for support of forming simulation.

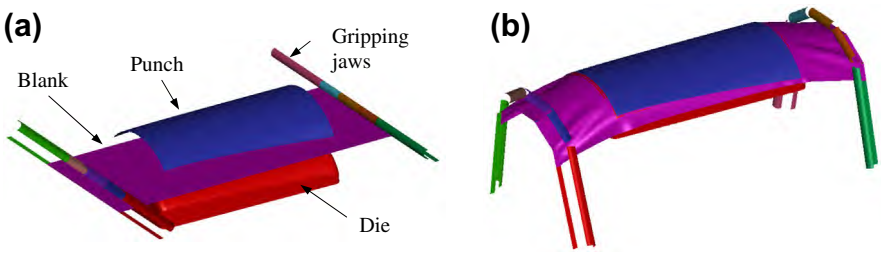


FIGURE 13.41

Stretch forming simulation, (a) initial setup, and (b) final stage performed in simulation.

13.5.1 CORE PANEL

The panel is made of aluminum sheet of 0.01 in. The part size is 90 in. × 33 in. × 0.28 in., as shown in Figure 13.42(a). The problem with the core panel is that the part reveals severe wrinkles around the corners of the concave pockets, as shown in Figure 13.42(b); this was first identified by the shop floor mechanists when the panel was physically formed. This case study illustrates the use of simulation software to learn the causes of the wrinkles and possible adjustment in process parameters to minimize wrinkles.

Simulations of three cases with binder force of 1, 10, and 100 tons (Cases 1, 2, and 3, respectively), were carried out. The setup is depicted in Figure 13.43. The results in Figure 13.44 show that wrinkles occur at the corner areas, as revealed in the physically formed part. In addition, the wrinkles reduce as the binder force increases. As we zoom in to take a closer look, for example, at one of the corner areas near the top center of the panel shown in Figure 13.45, it shows clearly that the strain points in two elements (A and B) gradually move out of the wrinkle area as binder force increases.

The physics behind the results is straightforward. When the binder force is increased, the blank is held tighter; therefore, moving the blank material is more restricted, especially around the corner areas where material tends to accumulate as it is being bent over to form the pockets. This phenomenon is

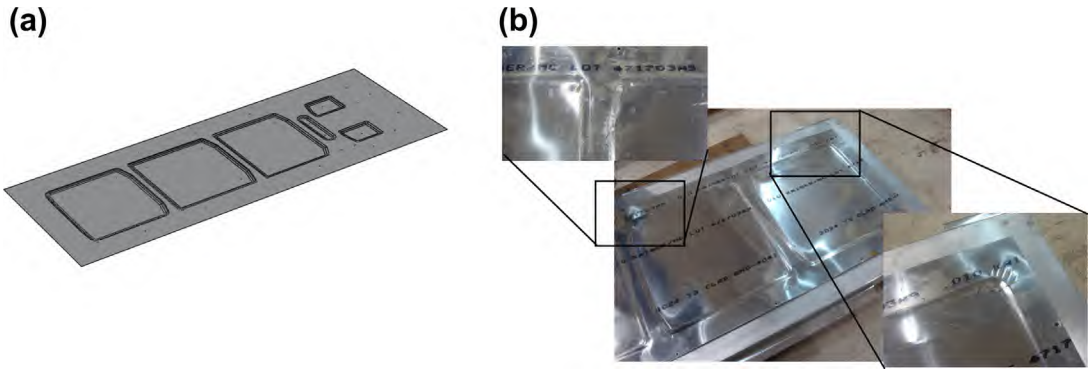


FIGURE 13.42

Core panel, (a) CAD model, and (b) physically formed part with severe winkles.

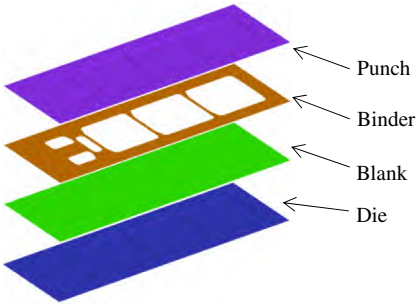


FIGURE 13.43

Simulation set up for the core panel example.

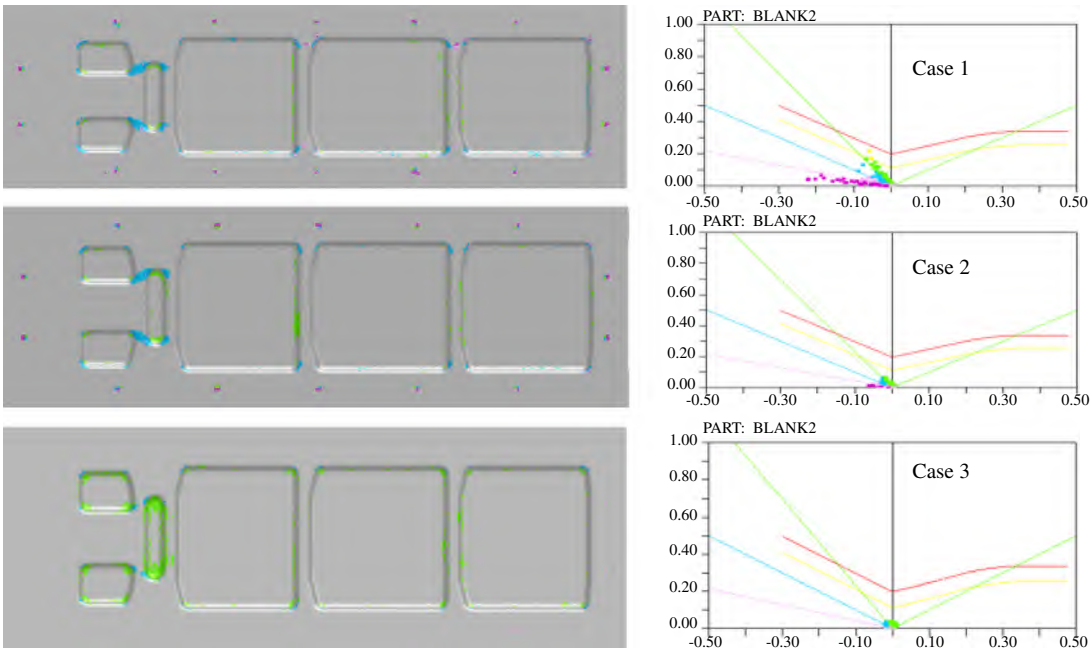


FIGURE 13.44

Forming simulations for varying binder force, Case 1: 1 ton, Case 2: 10 tons, and Case 3: 100 tons.

confirmed with the thickness distribution, as shown in Figure 13.46, around the same area. In addition, the displacement fringe plots shown in Figure 13.47 indicate similar conclusions. As shown in that figure, material movement is gradually reduced in both the vertical and horizontal directions as the binder force increases; this reveals the basic cause contributing to the diminishing of wrinkles in the corner areas.

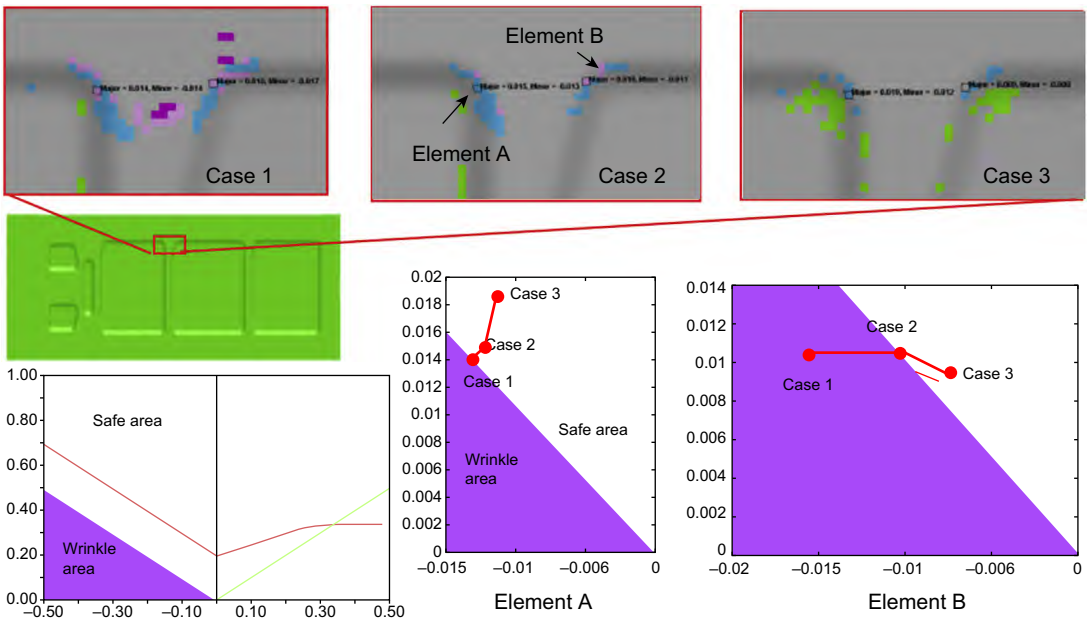


FIGURE 13.45

Strain points moving out of wrinkle area as binder force increasing.

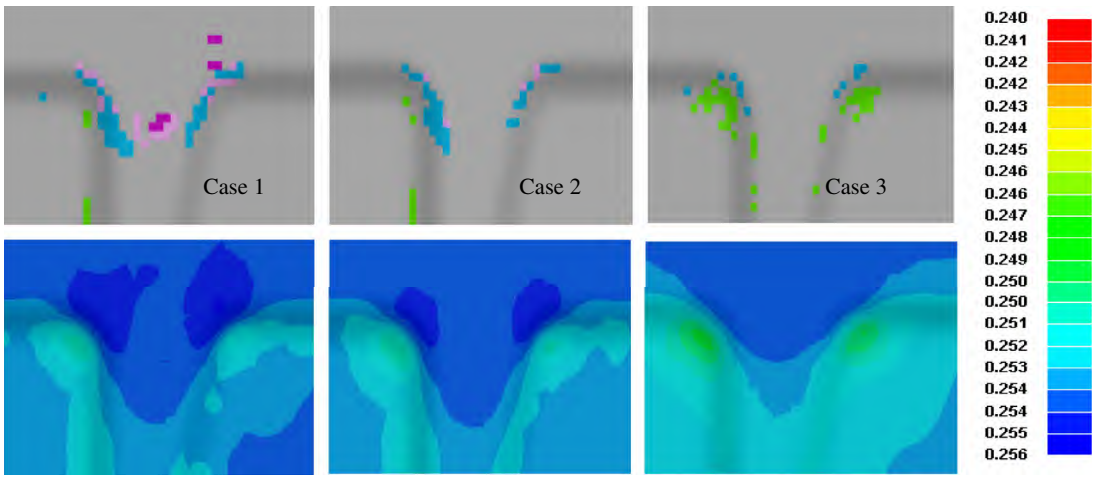


FIGURE 13.46

Thickness distributions in all three cases.

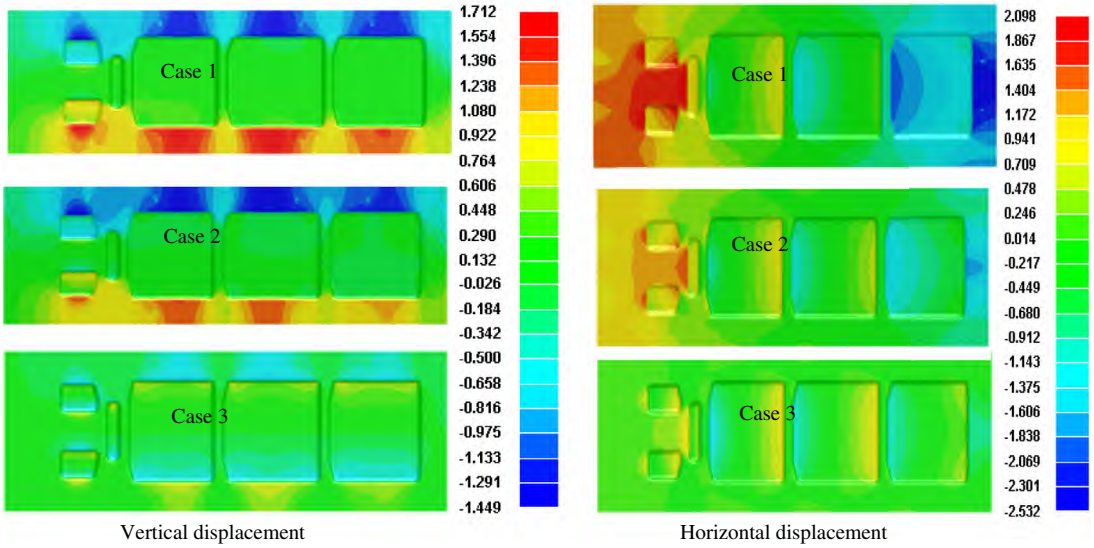


FIGURE 13.47

Displacement fringe plots.

13.5.2 WHEEL FAIRING

In this case study, we present the procedure of setting up the incremental draw forming analysis with DynaForm for a sample part using existing tooling and blank design. The sheet metal part (Figure 13.48) made of Aluminum 2024 (AL2024) is roughly 1178 mm long × 613 mm wide, with a pocket depth of 208 mm and a thickness of 0.81 mm. Instead of a step-by-step instruction, the aim of this study is a demonstration of software capabilities in modeling sheet forming processes. We recommend that interested readers download the DynaForm user manual from ETA's website (www.eta.com/dynaform) for details on using the software for various sheet forming applications.

Important steps for the setup of a general forming process in DynaForm are shown in Figure 13.49. Before creating the drawing simulation, we need to first bring in the tooling geometry. In this case, three tools are required to form the part; namely the die, the punch, and the binder (blank holder),

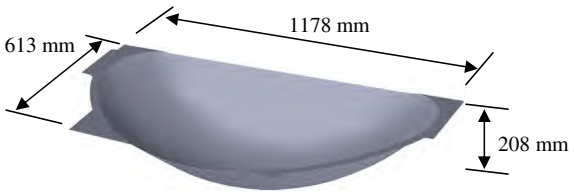


FIGURE 13.48

Sample part.

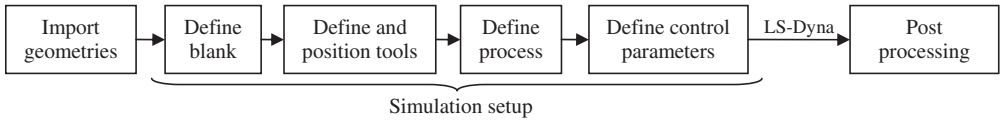


FIGURE 13.49

Flowchart of incremental forming simulation setup in DynaForm.

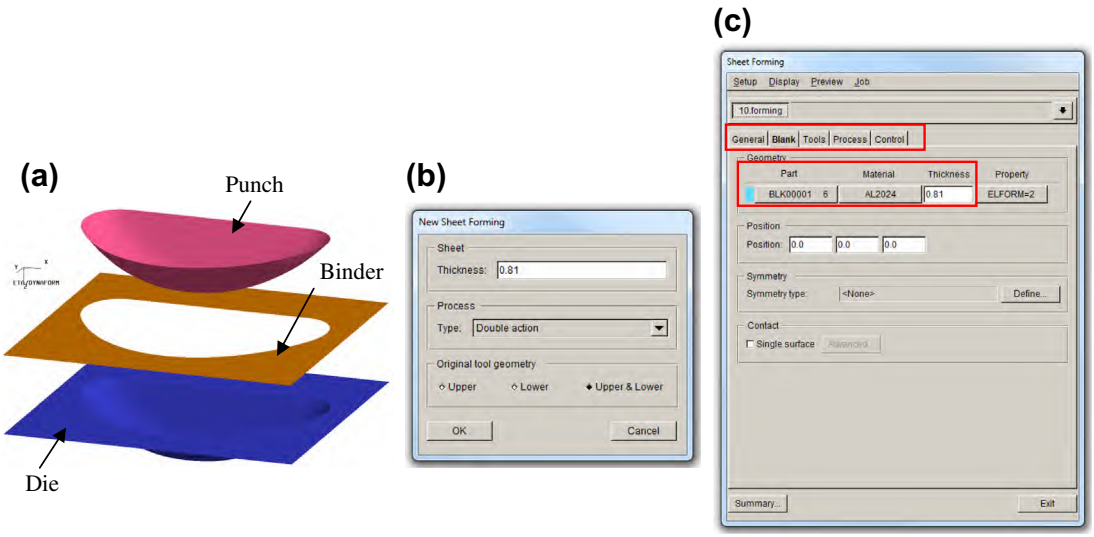


FIGURE 13.50

Setup a forming simulation, (a) imported tooling geometry, (b) New Sheet Forming dialog box, (c) Sheet Forming dialog box, *Blank* tab.

as shown in Figure 13.50(a). Note that instead of 3D solid models of the tooling blocks, only the tooling surfaces that will be in contact with the sheet during forming are needed.

Next, we start to create a new forming simulation by choosing *AUTO SETUP* > *SHEET FORMING* on the main menu. In the New Sheet Forming dialog box (Figure 13.50(b)), we define the sheet thickness to be 0.81 mm and the process type as *double action*. The sheet forming dialog box then appears, as shown in Figure 13.50(c), where the forming process can be constructed intuitively by going through the five tabs: *General*, *Blank*, *Tools*, *Process*, and *Control*.

After entering the basic task information, such as title and coordinate system, in the first tab *General*, we move on to the second tab, as shown in Figure 13.50(c), to define the blank. The geometry of the initial flat blank can be created by drawing a rectangle and then meshing the surface (Figure 13.51(a)). Alternatively, if the blank shape is pre-designed, we may also import it as a surface or a boundary curve. The blank material can be assigned by selecting an existing material from the DynaForm material library or by creating a new one using the material models provided. For this case, we will build our own material using the material model *3-Parameters Barlat* and the *Krupskowsky*

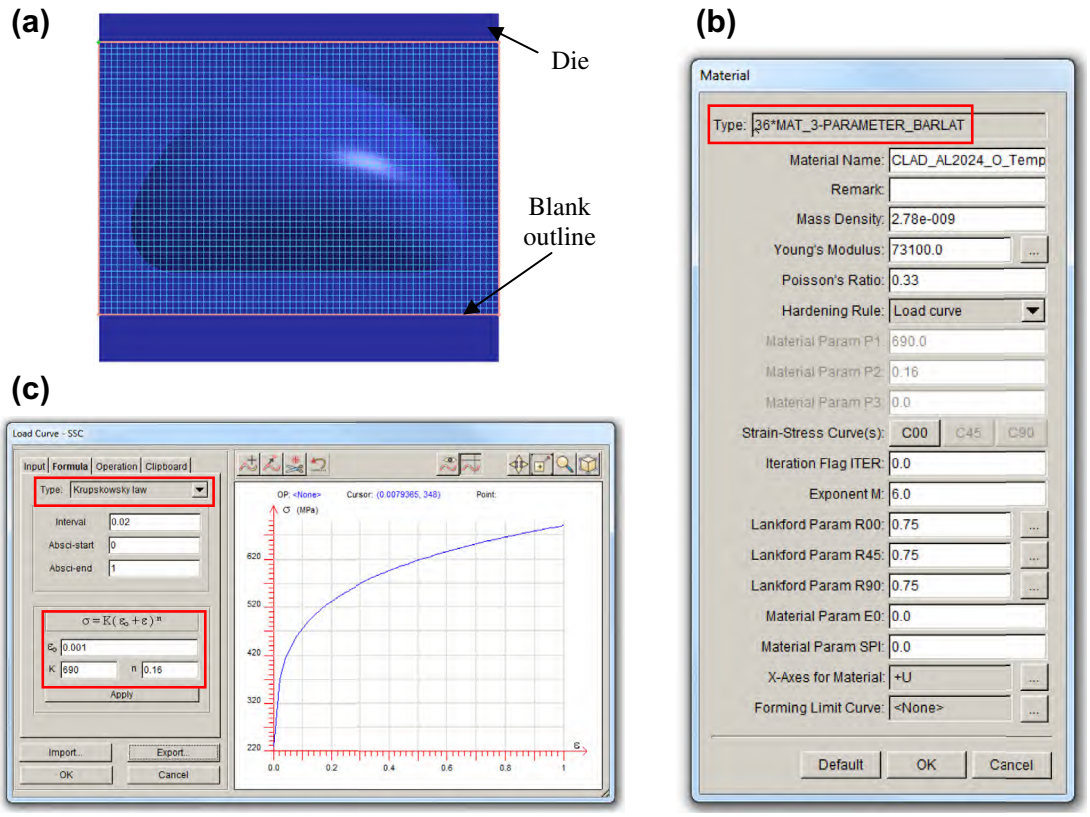


FIGURE 13.51

(a) Blank mesh generated based on a rectangular outline, (b) material parameters for *3-Parapeter Barlat* model for AL2024, (c) *Krupskowsky* strain-hardening curve for AL2024.

strain-hardening model (power law with pre-strain), as illustrated in Figures 13.51(b) and (c). Note that the material properties and parameters we use here are for demonstration purposes only and may not be as accurate.

Once the blank material is selected, we proceed to the next tab to define the forming tools. The tooling geometries we imported earlier need to be meshed first, and then linked to the three default tools (die, punch, and binder) for a double-action process listed on the left side, as shown in Figure 13.52(a). Note that it is possible to add other tools if necessary. The positions of the blank and tooling can be automatically determined according to the configuration right before starting the forming operation using the *positioning* function in Figure 13.52(a). In the meantime, a friction coefficient needs to be defined separately for each tool. In this case, we use a 0.17 friction coefficient to model the contact conditions between the blank and all tools (Figure 13.52(a)).

Next, in the *Process* tab, we define the motions of the tools during the sheet forming operation. As can be seen in Figure 13.52(b) for double-action drawing, two default subprocesses are listed (i.e.,

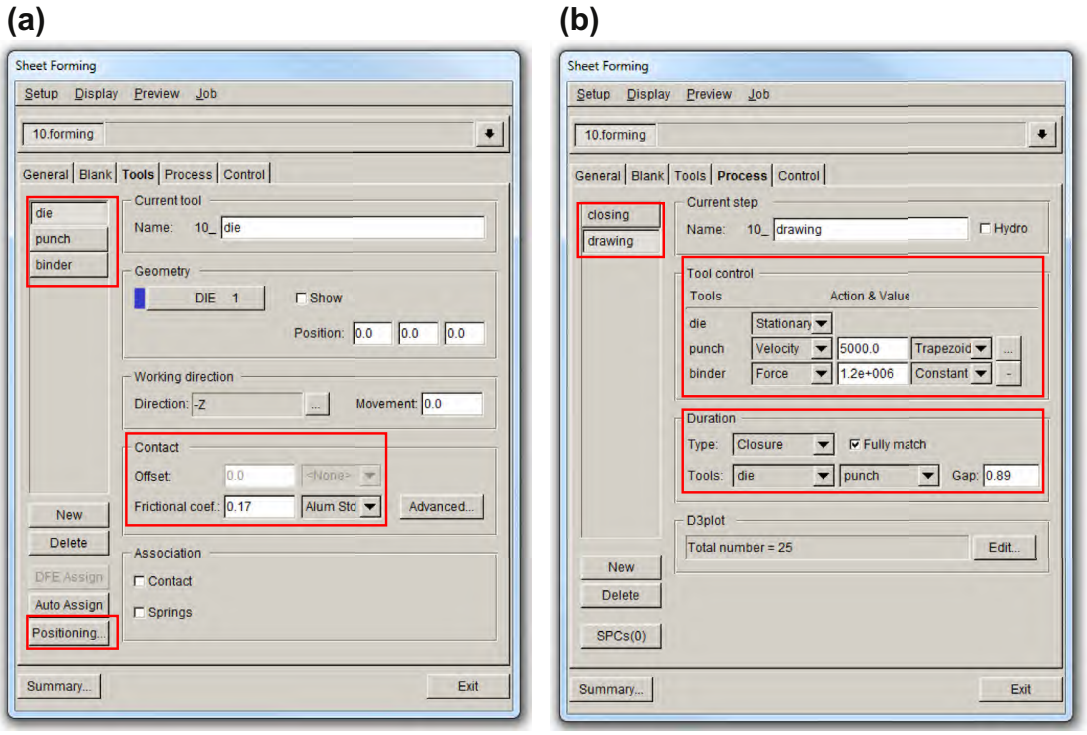


FIGURE 13.52

(a) Sheet forming window – *Tools* tab, (b) sheet forming window – *Process* tab.

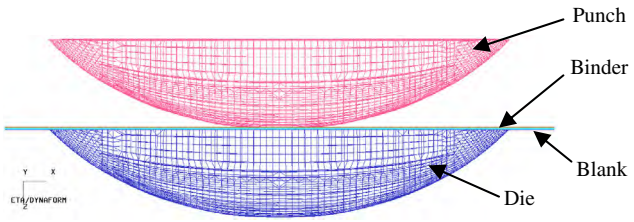


FIGURE 13.53

Automatically positioned blank and tooling. The punch and binder are right above the blank.

the closing of the binder against the die and the drawing process in which the punch deforms the blank). Again, additional subprocesses can be created before or after any existing process. In each subprocess, the action of individual tools can be specified in terms of either velocity, displacement, or force. For example, it is shown in Figure 13.53 that during the drawing process, while the die will stay still, the punch will travel at a speed of 5000 m/s, and a 1.2E6 N ($\sim$120 ton) force will be applied to the

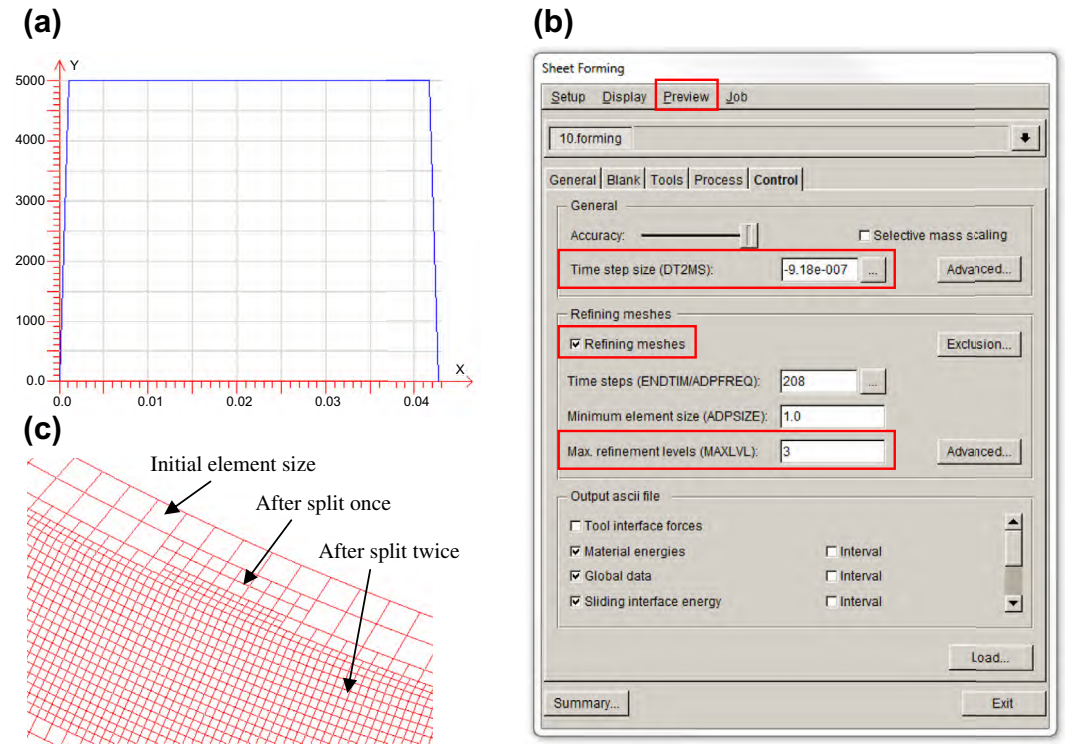


FIGURE 13.54

(a) Trapezoidal velocity curve for the punch, (b) sheet forming window – *Control* tab, (c) mesh refinement during incremental analysis.

binder. Figure 13.54(a) illustrates the automatically generated trapezoidal velocity curve for the punch, which can be edited by the user. In addition, the duration of each subprocess can be controlled using a prescribed time period. Alternatively, we may also choose to terminate the process when a tool has traveled for a specific distance or when two tools close with each other. As shown in Figure 13.52(b), the termination moment for the drawing process in this example is set to be the instant when the gap between the punch and the die becomes 0.89 mm, which is 10% larger.

The last step in the simulation setup is to adjust the solver control parameters in the *Control* tab. As shown in Figure 13.54(b), in this example we will accept the time-step size calculated by the software and set the mesh refinement level to be three; this means each element will split at most two times during the simulation, as illustrated in Figure 13.54(c). Once the setup is complete, we can click on *Preview* (Figure 13.54(b)) to review a kinematic animation of the forming process and verify the tooling motions. If everything looks correct, we then start the simulation by sending the job to LS-Dyna solver.

The computation time for running this incremental analysis was around 30 minutes on a Dell T7500 workstation with eight CPUs (threads). The simulation results can be accessed from the

post-processor named *eta/Post*, where important information (e.g., thickness, stress, strain distributions, and the forming limit diagram) can be displayed, as shown in Figures 13.31(a) and (b). Note that for this case, the formed part is acceptable in general, despite the minor wrinkling on the flat surface. A subsequent trimming operation can be simulated in DynaForm to obtain the part shape shown in Figure 13.48.

13.6 SUMMARY

In this chapter, we briefly introduced the subject of sheet metal forming simulation. We discussed the basic mechanics and computational methods that should help readers understand how sheet forming simulation is carried out. We devoted a major effort to discussing sheet forming process planning and tooling design using simulation technology and software tools. We included one-step simulation, die design with addendum, and drawbead design. We also described incremental forming analysis that shows the detailed steps in how the blank deforms during the forming process, as well as springback analysis that plays an important role in compensating die geometry. We hope readers come away with a general understanding of sheet forming simulation and are able to apply some of the concepts and ideas in their own examples. In addition to discussing forming simulation, we offer a review of major forming simulation software packages, which should provide a basic understanding of availability of software and their technical capabilities. With this information, readers should be able to make an adequate software selection decision when offered an opportunity. We also presented two case studies to provide more insight into using simulation for support of sheet forming design and manufacturing.

We hope that readers do see the importance and value of using computing simulation technology in the early product design stage to support part design and manufacturing using sheet forming processes. This chapter, hopefully, has opened a door for readers to further investigate and learn more about sheet forming theory, technology, and software tools. We encourage everyone to continue learning and using forming simulation tools, and to become competent engineers in dealing with complex design and manufacturing issues in metal forming.

QUESTIONS AND EXERCISES

13.1. Conduct a case study in the application of sheet forming technology and software in product development in the automotive or aerospace industry. In your one-page report, please include the following:

- **a.** Name of the company or organization
- **b.** Source of the information (article, paper, magazine, website, YouTube, etc.)
- **c.** What is the nature of the product?
- **d.** What challenges are engineers facing in product development?
- **e.** What are the advantages of using sheet forming technology for such applications? Discuss cost saving if possible.

13.2. As discussed in this chapter, hydroforming is an important forming process for the automotive industry. Please review the following youTube videos and answer questions below.

http://www.youtube.com/watch?v=n-ht_5Ysurc
http://www.youtube.com/watch?v=XUMs3cWBlks
http://www.youtube.com/watch?v=AojqlCCyGVc

a. What are the major advantages of hydroforming that you learned from these videos? Please list at least two.

b. What are some of the typical parts that can be made best by hydroforming, and why?

13.3. We discussed a few forming simulation software tools in this chapter. There are other tools commercially available, such as Form-Advisor. Please review the following youTube video to obtain a good understanding in this software.
http://www.youtube.com/watch?v=IskACmU4KIA)
Find another forming simulation software that you may learn by, for example, reviewing a youTube video, technical paper, case study, software website, or software brochure. Provide a short description of the software, and compare its capabilities with those of Form-Advisor.

REFERENCES

Altair Engineering, 2011. HyperWorks 11 Help Document.

Altan, T., Tekkaya, A.E., 2012. Sheet Metal Forming Fundamentals. ASM International.

Banabic, D., 2010. Sheet Metal Forming Processes. Springer.

Banabic, D., Balan, T., Comsa, D.S., 2000. A new yield criterion for orthotropic sheet metals under plane-stress conditions. Proceedings of the 7th Conference "TPR2000", Cluj Napoca, Romania, 217–224.

Barlat, F., Lian, J., 1989. Plastic behavior and stretchability of sheet metals (Part I): A yield function for orthotropic sheet under plane stress conditions. International Journal of Plasticity 5, 51–56.

Barlat, F., Becker, R.C., Hayashida, Y., Maeda, Y., Yanagawa, M., Chung, K., Brem, J.C., Lege, D.J., Matsui, K., Murtha, S.J., Hattori, S., 1997a. Yielding description for solution strengthened aluminum alloys. International Journal of Plasticity 13, 185–401.

Barlat, F., Maeda, Y., Chung, K., Yanagawa, M., Brem, J.C., Hayashida, Y., Lege, D.J., Matsui, K., Murtha, S.J., Hattori, S., Becker, R.C., Makosey, S., 1997b. Yield function development for aluminum alloy sheets. Journal of the Mechanics and Physics of Solids 45, 1727–1763.

Barlat, F., Aretz, H., Yoon, J.W., Karabin, M.E., Brem, J.C., Dick, R.E., 2005. Linear transformation-based anisotropic yield functions. International Journal of Plasticity 21, 1009–1039.

Bendsøe, M., Sigmund, O., 2003. Topology Optimization: Theory, Methods and Applications. Springer.

Cazacu, O., Barlat, F., 2001. Generalization of Drucker's yield criterion in orthotropy. Mathematics and Mechanics of Solids 6, 613–630.

Engineering Technology Associates, Inc, 2011. eta/DYNAFORM User's Manual, Version 5.8.1.

ESI Group, 2011. PAM-STAMP 2G 2011 User's Guide.

Firat, M., 2004. Sheet metal springback prediction including initial plastic anisotropy. In: Akkok, M., Kaftanoglu, B. (Eds.), Proceedings of the 3rd International Conference on Design and Production of Dies and Molds. Dies and Molds Metu, Bursa, pp. 248–254.

Firat, M., 2007. U-Channel forming analysis with an emphasis on springback deformation. Materials and Design 28, 147–154.

Goodwin, G.M., 1968. Application of strain analysis to sheet metal forming problems in the press shop. Society of Automotive Engineers. No. 680093, 380–387.

Hershey, A.V., 1954. The plasticity of an isotropic aggregate of anisotropic face centered cubic crystals. Journal of Applied Mechanics 21, 241–249.

Hill, R., 1948. A theory of the yielding and plastic flow of anisotropic metals. Proceedings of the Royal Society London A 193, 281–297.

Hill, R., 1979. Theoretical plasticity of textured aggregates. Mathematical Proceedings of the Cambridge Philosophical Society 85, 179–191.

Hill, R., 1987. Constitutive dual potential in classical plasticity. Journal of the Mechanics and Physics of Solids 35, 23–33.

Hill, R., 1990. Constitutive modelling of orthotropic plasticity in sheet metals. Journal of the Mechanics and Physics of Solids 38, 405–417.

Hill, R., 1993. A user-friendly theory of orthotropic plasticity in sheet metals. International Journal of Mechanical Sciences 15, 19–25.

Kalpakjian, S., Schmid, S.R., 2010. Manufacturing, Engineering & Technology, sixth ed. Prentice Hall.

Keeler, S.P., 1961. Plastic instability and fracture in sheet stretched over rigid punches, PhD Thesis. Massachusetts Institute of Technology, Boston.

Kobayashi, S., Oh, S., Altan, T., 1989. Metal Forming and the Finite-Element Method. Oxford University Press.

Lankford, W.I., Snyder, S.C., Bauscher, J.A., 1950. New criteria for predicting the press performance of deep-drawing sheets. Transaction ASM 42, 1196–1232.

Levy, M., 1870. Memoire sur les equations generales des mouvements interieurs des corps solides ductiles au dela des limites oil 1'elasticite pourrait les ramener a leur premier etat. Comptes Rendus Academie des Sciences, Paris, 70, pp. 1323–1325.

Logan, R., Hosford, W.F., 1980. Upper-bound anisotropic yield locus calculations assuming (111)-Pencil glide. International Journal of Mechanical Sciences 22, 419–430.

Makinouchi, A., 1996. Sheet Metal Forming Simulation in Industry. Journal of Materials Processing Technology 60, 19–26.

Marciniak, Z., Duncan, J.L., Hu, S.J., 2002. Mechanics of Sheet Metal Forming, second ed. Butterworth-Heinemann.

Palaniswamy, H., Billur, E., 2012. Plastic Deformation-Flow Stress (Chapter 4), Anisotropy, and Formability. In: Altan, T., Tekkaya, A.E. (Eds.), Sheet Metal Forming Fundamentals. ASM International.

Palaniswamy, H., 2012. Plastic deformation-state of stress, yield criteria flow rule, and hardening rules (Chapter 5), Anisotropy, and Formability. In: Altan, T., Tekkaya, A.E. (Eds.), Sheet Metal Forming Fundamentals. ASM International.

Tang, A., Lee, W.C., Pierre, S., He, J., Liu, K., Chen, C.C., 2005a. CAE Based Die Face Engineering Development to Contribute to the Revitalization of the Tool & Die Industry. AIP Conference Proceedings 778 (1), 50–59.

Tang, A., Lee, W.C., He, J., Xu, J., Liu, K., Chen, C.C., 2005b. Die Face Engineering based Springback Compensation Strategy and Implementation. AIP Conference Proceedings 778 (1), 314–321.

Tekkaya, A.E., 2000. State-of-the-art of simulation of sheet metal forming. Journal of Materials Processing Technology 103, 14–22.

Tisza, M., Lukács, Zs, Gál, G., 2008. Integrated process simulation and die-design in sheet metal forming. International Journal of Material Forming 1, 185–188.

Tisza, M., 2004. Numerical modeling and simulation in sheet metal forming. Journal of Materials Processing Technology 151, 58–62.

Tresca, H., 1864. On the yield of solids at high pressures. Comptes Rendus Academie des Sciences, Paris, 59, pp. 754 (translated from French).

Vegter, H., Drent, P., Huetink, J., 1995. A planar isotropic yield criterion based on material testing at multi-axial stress state. In: Shen, S.F., Dawson, P.R., Balkema, A.A. (Eds.), Simulation of materials processing-theory: methods and applications, pp. 345–350.

von Mises, R., 1913. Mechanics of solids in plastic state. Göttinger Nachrichten Mathematical Physics 4, 582–592 (translated from German).

Wagoner, R.H., Chenot, J.-L., 2001. Metal Forming Analysis. Cambridge University Press.

Zhou, D., Wagoner, R.H., August 30, 1993. Development and Applications of Sheet Forming Simulation. NUMISHEET '93. Tokyo, Japan 3–17.

CHAPTER

RAPID PROTOTYPING 14

CHAPTER OUTLINE

14.1 Introduction 745
14.2 RP Process and Tutorial Example 746
14.2.1 General Process 746
14.2.2 Engine Block Example 747
14.3 Rapid Prototyping Systems 750
14.3.1 Liquid-Based Systems 750
14.3.2 Solid-Based Systems 751
14.3.3 Powder-Based Systems 752
14.4 Advanced RP Systems 754
14.4.1 Solidica 754
14.4.2 Electron Beam Melting 755
14.4.3 Laser Engineered Net Shaping 757
14.4.4 Micro-Manufacturing RP Systems 757
14.5 Rapid Prototyping Applications 759
14.5.1 Design Applications 760
14.5.2 Manufacturing Applications 761
14.5.3 Art Applications 761
14.5.3.1 Art Design 762
14.5.3.2 Museum Application 762
14.5.3.3 Props 763
14.5.4 Medical Applications 763
14.5.4.1 Presurgery Planning and Rehearsal 763
14.5.4.2 Education and Research Application 766
14.5.4.3 Dental Applications 767
14.5.5 Bioengineering Applications 768
14.5.5.1 Custom Prosthesis and Implantation 768
14.5.5.2 Scaffolds for Tissue Engineering 769
14.5.5.3 Organ Printing 771
14.5.6 Personal RP 771
14.5.7 Other Applications 771

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00014-4
Copyright © 2015 Elsevier Inc. All rights reserved.

14.6 Case Study: RP for Complex Assembly 773
14.6.1 Single-Piston Engine 773
14.6.2 Formula SAE Racecar 775
14.6.2.1 Scale Factor 775
14.6.2.2 Model Modification 777
14.6.2.3 Model Conversion 780
14.6.2.4 Model Fabrication 780
14.7 Summary 783
Questions and Exercises 783
References 784

Rapid prototyping (RP), also called 3D printing or *solid freeform fabrication* (SFF), refers to the technology and apparatus that support fabrication of physical objects directly from solid models created in CAD using additive layer manufacturing techniques without manufacturing process planning, tooling, or fixtures. The development of RP technology started in the 1970s with the first research article published in 1977 (Swainson, 1977). The first apparatus for rapid prototyping became commercially available in the late 1980s. The first commercial machine was shipped in 1988 by 3D Systems, Inc., which was a stereolithography apparatus (SLA) using photopolymer as building material. The earlier RP machines were used to produce physical models and prototype parts; today, they are used for a much wider range of applications and are even used to manufacture production-quality parts in a small quantity. This technology has the potential to reduce the turnaround time in product design and development.

Various kinds of RP machines are commercially available. In general, they can be categorized as solid-, liquid-, or powder-based systems, depending on the materials they use. All are capable of producing relatively durable objects suitable for support of product design and development. A few emerging technologies are capable of fabricating metal objects that are as strong as machined or cast parts; e.g., EBM, electron beam melting. EBM uses an electron beam to melt titanium powder, resulting in dense and high-strength titanium objects. These metal parts are extremely suitable for functional prototyping and small-volume production. Some RP technologies have been extended to support microscopic manufacturing. The need for micro parts is largely seen in microelectronics as well as in the optoelectronics fabrication industries. They are required mostly for MEMS (micro-electro-mechanical systems) and medical applications.

As a powerful and flexible tool for product manufacturing, rapid prototyping has been used extensively in support of product development, providing considerable progress toward improvements in product development time, cost, and product quality. In this chapter, several major areas of rapid prototyping application will be introduced, including design, manufacturing, art, medical science, bioengineering, and others.

Fabricating physical prototypes of large-scale and complex assemblies using Dimension 1200sst of Stratasys, Inc., is included in this chapter as case studies that provide more in-depth discussion in using RP. Two examples are included: a single piston engine and a Formula SAE racecar.

The aim of this chapter is to provide readers with an introduction to RP technology. Overall objectives include: (1) provide readers a general understanding on RP technology and the various machines available commercially, (2) help readers become more familiar with emerging RP and their

applications in micro-manufacturing and other fields, and (3) offer case studies so that readers can apply the same principles and methods for their own applications.

14.1 INTRODUCTION

For those who watched the movie *Jurassic Park III*, you probably remember the resonating chamber that was fabricated by a paleontology student Billy Brennan, Dr. Alan Grant's graduate student, using a machine (see www.anyclip.com/movies/jurassic-park-iii/resonatingchamber-prototype). The geometry of the resonating chamber was captured by scanning the interior of a velociraptor skull, and then slicing it into thousands of 2D layers. Geometric data captured in the 2D layers were supplied to the machine that fabricated a physical replicate of the resonating chamber. When air passed through the chamber, it replicated the sounds that would have been made while the animal was alive. Storing the resonating chamber in his bag, Dr. Grant later used it against the raptors after the stolen eggs were returned. The resulting sounds confused the dinosaurs, and they exited with their eggs after hearing the approaching helicopter.

How does the machine work? How was the chamber's scanned image converted into a 3D physical model that replicated the sounds? The technology is called rapid prototyping (RP), 3D printing, or solid freeform fabrication (SFF). Rapid prototyping machines are mechanical devices used to convert three-dimensional, computer-generated designs into physical prototypes. Such an apparatus permits the production of highly accurate models with levels of detail and accuracy that exceed those typical of traditional casts. Today, RP plays an important role in product design and prototyping, and it contributes to shortening the overall product development cycle, improving product quality, and reducing cost.

In general, fabrication processes fall into three categories: subtractive, additive, and compressive (Jacobs, 1995). In a subtractive process, a block of material is carved out to produce the desired shape. An additive process builds an object by joining particles or layers of raw material. A compressive process forces a semisolid or liquid material into the desired shape, in which it is then induced to harden or solidify. Most conventional fabrication processes fall into the subtractive category; these include machining processes such as milling, turning, and grinding. Machining methods are difficult to use on parts with very small internal cavities or complex geometry, so as the compressive processes, which are also conventional, including forming (or stamping), casting, and molding. All these processes require manufacturing process planning, tooling, and fixture design and manufacturing. They often take days, if not weeks or even months, to complete.

The newly developed RP technology employs an additive process that replicates physical parts from their digital mockup in CAD data through layer manufacturing. The 3D printers allow designers to quickly create physical prototypes of their designs, rather than just two-dimensional (2D) pictures or virtual mockup in CAD. Such models have numerous uses; for example, they make excellent visual aids for communicating ideas to coworkers or customers. Prototypes can be used for design testing or verification. For example, an aerospace engineer might mount a model airfoil in a wind tunnel to measure lift and drag forces. Design engineers have always used prototypes to aid product design; RP allows them to be made faster and less expensively.

In addition to prototypes, RP technology can be used to make tooling (referred to as *rapid tooling*) and even production-quality parts (called *rapid manufacturing*). For small production runs and complicated objects, rapid prototyping is often the best manufacturing process available. Most

prototypes require from hours to days to build, depending on the size and complexity of the object. This may seem slow, but it is much faster than the weeks or months required to make a prototype by traditional means (e.g., forming or casting). These dramatic time savings allow manufacturers to bring products to market faster and for less cost. In 1994, Pratt & Whitney achieved "an order of magnitude [cost] reduction [and] … time savings of 70 to 90 percent" by incorporating rapid prototyping into their investment casting process (Aronson, 2000).

In this chapter, we will start by briefly reviewing the general RP process, from CAD to physical prototype, as well as major technology and commercially available machines. Advanced RP technology that is capable of fabricating high-strength metal parts will be discussed, and then a number of RP technologies extended for micro-manufacturing will be introduced. After introducing the technology, we will discuss general RP applications in various industry and research sectors. We will wrap up this chapter by introducing two practical examples as case studies: a single-piston engine and a Formula SAE racecar.

14.2 RP PROCESS AND TUTORIAL EXAMPLE

This section will outline a general process of RP, including key elements and basic skills in using rapid prototyping. A single-piston engine block shown in Figure 14.1 is included as a tutorial sample to illustrate various steps involved in RP.

14.2.1 GENERAL PROCESS

In general, five steps are involved in the RP process, as shown in Figure 14.1. They are CAD solid modeling, model conversion to STL, STL model slicing, model fabrication, and post-processing, resulting in a physical prototype.

The process starts with a valid solid model in CAD that represents the part design. The designer can use a pre-existing CAD file or may wish to create one expressly for prototyping purposes. In some cases, an existing CAD model may need a few modifications to support RP—for example, cut off a portion of the cover to reveal the interior structure or adding support to a delicate part.

CAD packages use different algorithms to represent solid objects. To establish consistency, the STL (STereoLithography) format (Chua et al., 2010) has been adopted as the standard by the rapid

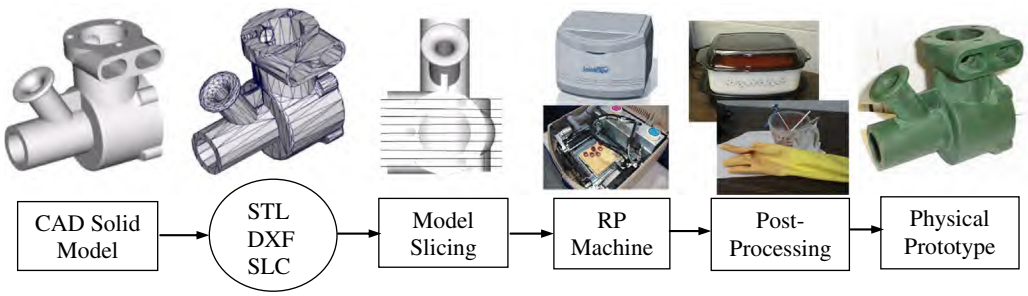


FIGURE 14.1

The general RP process.

prototyping industry. The second step, therefore, is to convert the CAD file into STL format. This format represents 3D boundary geometry of an object as an assembly of triangular facets. The file contains the coordinates of the vertices and the direction of the outward normal of each triangle. Because STL files use triangular facets, they cannot represent curved surfaces exactly. Increasing the number of triangles improves the approximation but at the cost of bigger file size. Large, complicated files require more time to process, so the designer must balance accuracy with file size to produce a useful and adequate STL file. Since the *.stl* format is universal, this process is identical for all of the RP build techniques. Note that some RP machines accept additional file formats, such as the SLC (StereoLithography Contour), which employs a 2-½D contour representation of a CAD model. It consists of successive cross-sections taken at ascending Z intervals in which solid material is represented by interior and exterior boundary polylines (Chua et al., 2010).

The STL model is then brought into the slicing program that is usually proprietary and comes with the RP machine. The slicing software allows the user to adjust the size, location, and orientation of the model. Build orientation is important for several reasons. First, properties of physical parts fabricated using RP vary from one coordinate direction to another. For example, prototypes are usually weaker and less accurate in the Z (vertical) direction than in the X-Y plane. In addition, part orientation partially determines the amount of time required to build the model. Placing the shortest dimension in the Z direction reduces the number of layers, thereby shortening build time in general. The preprocessing software slices the STL model into a number of layers, usually from 0.001 in. (0.0254 mm) to 0.01 in. (0.254 mm) thick, depending on the build technique. The program also may generate an auxiliary structure to support the model during the build. Supports are useful for delicate features such as overhangs, internal cavities, and thin-walled sections.

Once the STL model is successfully sliced into thin layers, in which tool patterns are generated for both build and support areas for each layer, the physical model will then be fabricated using one of several build techniques (discussed in Section 14.3). RP machines build one layer at a time from polymers, wax, ABS (acrylonitrile butadiene styrene) filament, liquid resin, or ceramic or metal powders. Most machines are fairly autonomous, needing little intervention.

The final step is post-processing, which involves removing the prototype from the machine and detaching any supports. Some photosensitive materials need to be fully cured before use. Prototypes also may require minor cleaning and surface treatment. Sanding, sealing, and/or painting the model will improve its appearance and durability.

14.2.2 ENGINE BLOCK EXAMPLE

The engine block of a single-piston engine shown in Figure 14.2(a) is employed to further illustrate the RP process. The enclosing envelope of the block is about 3 in. × 2 in. × 3 in. (76.2 mm × 50.8 mm × 76.2 mm). This engine block was printed using a Solidscape 3D printer, shown in Figure 14.3 (*www.solid-scape.com*), which is primarily used to produce wax-like patterns for lost-wax casting (also called investment casting) and mold-making applications. Solidscape RP machines are widely adopted by the jewelry industry due to their extremely high printing resolutions.

The CAD model of the block was first exported as STL from Pro/ENGINEER with a default chord height of 0.0160 in. (0.407 mm). A chord height that determines the accuracy of the STL model measures the maximum normal distance between the part boundary surface and triangle facet. A smaller chord height yields a more refined and accurate STL model with more triangles. In this

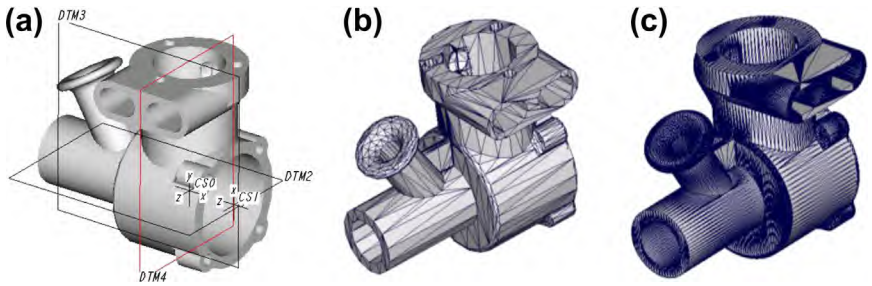


FIGURE 14.2

The single-piston engine block: (a) CAD solid model in Pro/ENGINEER (b) coarse STL model, and (c) fine STL model.

example, a default chord height yields a coarse STL model of 2,354 triangles (116 KB in file size), as shown in Figure 14.2(b). Note that this STL model is too coarse. Once printed, some of the facets are easily recognized, which is not desirable.

A smaller chord height of 0.000591 in. (0.015 mm), a minimum value provided in Pro/ENGINEER for this example, yields a much more refined STL model (Figure 14.2(c)), consisting of 17,524 triangles (856 KB in file size). Note that this STL model is sufficiently refined. Any smaller chord height that leads to a more refined model with more triangles may not be necessary due to the limitation of the printing resolution of the machine. For a Solidscape machine, the printing resolution can be up to 5000 × 5000 × 8000 dpi in the *X*-, *Y*-, and *Z*-axes, respectively, which is better than many other RP machines.

The STL model was brought into *ModelWorks*, a proprietary software that comes with the Solidscape RP machine, for slicing. The model was first positioned and oriented at a corner of a 6 in. × 12 in. (152.4 mm × 305.8 mm) build substrate (Figure 14.4(a)). Note that the *Z*-direction (vertical upward) as well as the layer thickness, in this case 0.001 in. (0.0254 mm), largely determined the overall model printing time. A typical tool pattern of an intermediate layer is shown in Figure 14.4(b),

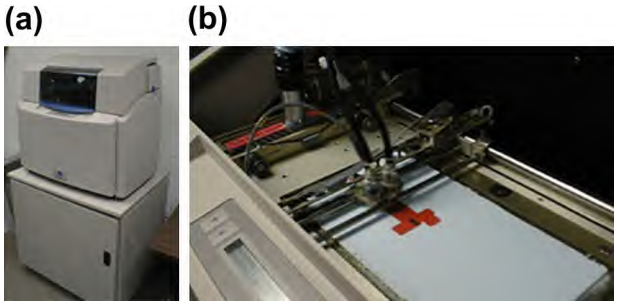


FIGURE 14.3

The Solidscape RP machine: (a) exterior view and (b) interior work zone.

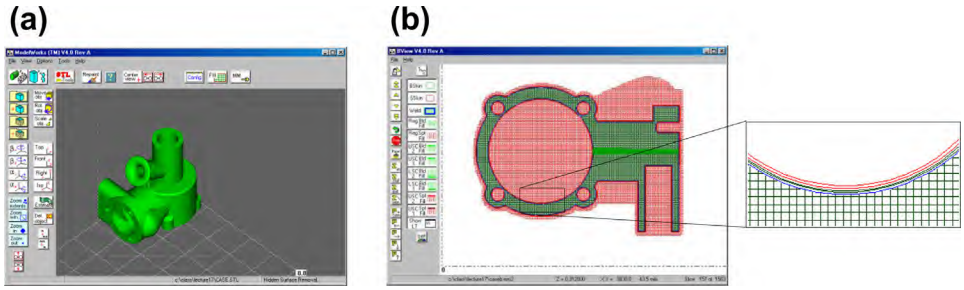


FIGURE 14.4

The *ModelWorks* slicing software: (a) model position and orientation and (b) toolpaths generated for an intermediate layer (this figure is reproduced in color on the book's companion website: http://booksite.elsevier.com/9780123820389).

in which green and red cross-hatch patterns indicate toolpaths of build and support materials, respectively.

The inkjet printing process, as implemented by Solidscape, begins with the build material (thermoplastic) and support material (wax) being held in a melted state inside two heated reservoirs. These materials are each fed to an inkjet print head that moves in the *X*-*Y* plane and shoots tiny droplets to the required locations to form one layer of the part, as shown in Figure 14.5. Both the build material and support material instantly cool and solidify. After a layer has been completed, a milling head moves across the layer to smooth the surface and remove extra materials. The particles resulting from this cutting operation are vacuumed away by the particle collector. The elevator then lowers the build platform and the part so that the next layer can be built.

After this process is repeated for each layer and the part is complete, the part is removed from the work chamber and put into an oven for 12 hours at 40°C to relieve the residue stress in the model.

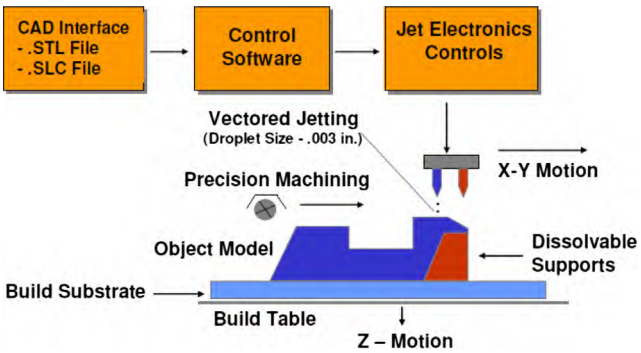


FIGURE 14.5

A schematic of the inkjet printing implemented in Solidscape RP machine (www.intriguity.com/images/solid2.jpg).

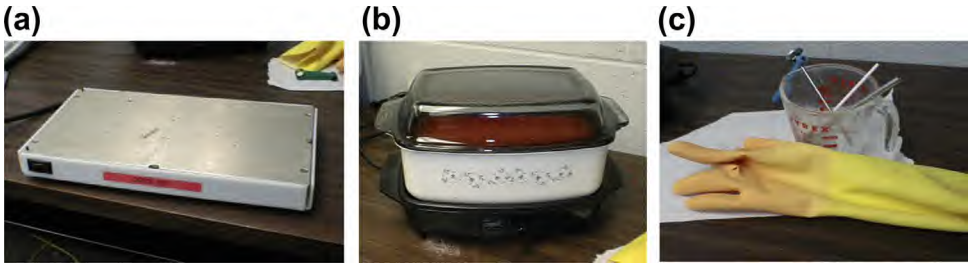


FIGURE 14.6

The postprocessing: (a) substrate removal, (b) wax dissolving, and (c) tools for remaining wax removal.

Afterward, the model with the substrate is placed on a hot plate (Figure 14.6(a)) heated to 80°C–100°C to remove the substrate. Then, the physical model is soaked in VSO BIOACT solvent (*www.cadbludental.com*) at 40°C for 1 hour (Figure 14.6(b)), and gradually heated VSO and the physical model from 40°C–65°C for 1 hour as wax is melted and dissolved. The remaining wax is removed using simple tools (e.g., brush and spatula), as shown in Figure 14.6(c). After the support material is completely removed, the final physical model of the engine block shown in Figure 14.1 (*right*) is ready for use.

14.3 RAPID PROTOTYPING SYSTEMS

There are a wide variety of commercially available RP technologies and systems. They can be categorized into three primary types according to the state—liquid-, solid-, and powder-based—in which their material begins.

14.3.1 LIQUID-BASED SYSTEMS

Liquid-based RP systems begin with model and/or support material in liquid form. One representative system and in fact the first system commercially available is StereoLithography Apparatus (SLA) from 3D Systems, Inc. (*www.3dsystems.com*).

The SLA machines build 3D parts by scanning photocurable (photosensitive or photopolymer) liquid resins through laser light (or UV: ultraviolet light), as illustrated in Figure 14.7(a). An ultraviolet laser cures a liquid resin into very thin layers, including interior and exterior cavities. Liquid resin is solidified due to heat radiated from the laser beam. This solidification process is referred to as *polymerization*. Liquid is replenished and leveled between layers. Support structures are also fabricated in the same way, but usually with a sparse microstructure. Completed parts require cleaning and UV or thermal postcuring.

Several SLA systems are offered by 3D Systems, including SLA Viper, iPro 8000, iPro 8000 MP, iPro 9000, and iPro 9000XL, with the price ranging between about $180K and $950K. The iPro 9000 XL shown in Figure 14.7(b) is the largest stereolithography platform commercially available. It can build parts as large as 59.1 in. × 29.53 in. × 21.65 in. (1500 mm × 750 mm × 550 mm). Entire dashboards (Figure 14.7(c)) or complete bumpers can be built in one piece.

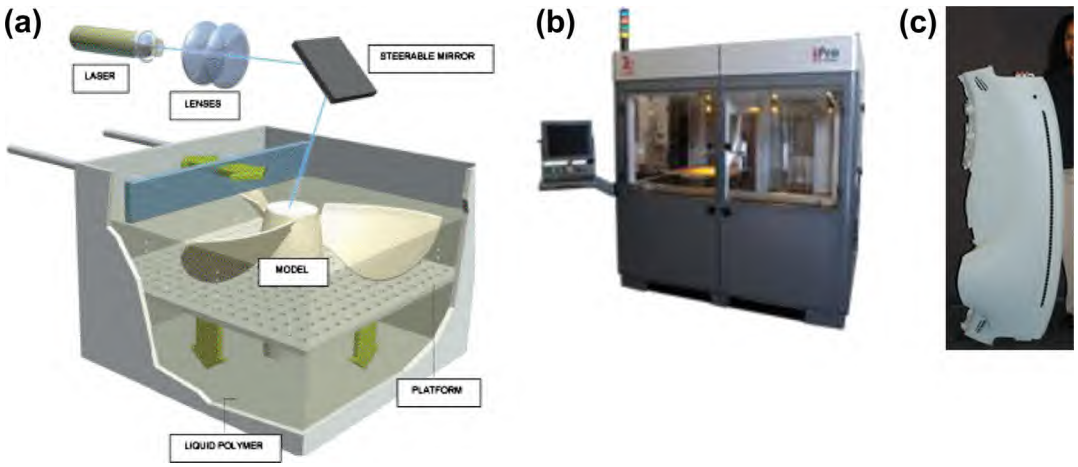


FIGURE 14.7

The StereoLithography apparatus: (a) part-building process (http://intech-ind.com/blog), (b) the iPro 9000 XL system (www.3dsystems.com), and (c) dashboard built by using iPro 9000 XL (www.makepartsfast.com/2011/10/2263/2011-make-parts-fast-handbook-stereolithography/).

SLA systems offer accuracy up to 0.0025 in. (0.0635 mm) with vertical resolution 0.00004 in. (0.001 mm), both are the best on the market. Typical layer thickness is 0.001 in. (0.0254 mm). One of the drawbacks of SLA is that the liquid resin is toxic and relatively lengthy postprocessing is usually required, in addition to relatively high unit costs. Besides the SLA systems of 3D Systems, other companies offer similar liquid-based RP systems. These include Polyjet of Objet Geometries (*www.objet.com*), D-MEC's Solid Creation System (SCS) (*www.d-mec.co.jp*), just to name two.

14.3.2 SOLID-BASED SYSTEMS

Solid-based manufacturing methods begin with materials that are most commonly in the form of a wire, sheet, or roll. Of these, Fused Deposition Modeling (FDM) is one of the most popular and widely used technologies; it uses materials, such as acrylonitrile butadiene styrene (ABS) polymer filament, to build parts. Both build material and soluble support material are unwound from cartridges (Figure 14.8(c)) then heated and extruded through nozzles that can switch material flows. The nozzles are heated to raise the temperature for the material just below its melting point and can be moved in horizontal directions by a numerically controlled mechanism, as illustrated in Figure 14.8(a). The physical model is fabricated by extruding small beads of thermoplastic material to form, layer by layer, as the material hardens immediately after extrusion from the nozzles.

One representative system employing FDM is Dimension 3D Printers from Stratasys (*www.stratasys.com*)—for example, Dimension 1200sst shown in Figure 14.8(b). The work envelope of the 1200sst is 10 in. × 10 in. × 12 in. (254 mm × 254 mm × 305 mm) with layer thickness 0.010 in. (0.254 mm). The price of Dimension 3D Printers range from about $15K (Dimension uPrint Plus) to $30K (Dimension Elite).

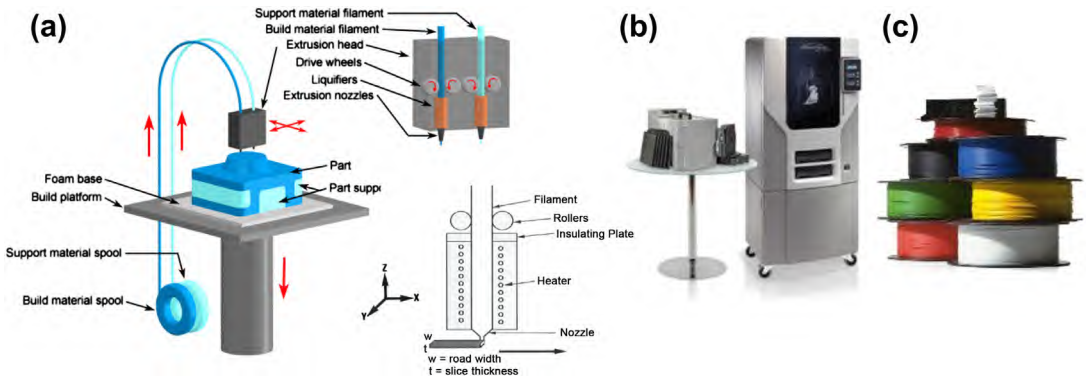


FIGURE 14.8

The Fused Deposition Manufacturing: (a) the build process (www.custompartnet.com/wu/fused-deposition-modeling), (b) the Dimension SST 1200sst (www.dimensionprinting.com/3d-printers/3d-printing-1200sst.aspx), and (c) ABS filament wound in cartridges.

In general, FDM systems produce parts with better strength; some of them are suitable for functional prototyping. However, the surface finish of the parts is relatively rough due to the thicker layers. In addition to FDM, other RP systems employ solid materials in various forms. These, among others, include Solidscape's Benchtop System that uses plastic and wax pallets for build and support, Cubic Technologies' Laminated Object Manufacturing (LOM) (*www.cubictechnology.com*) that uses paper or plastic sheets as materials, and 3D Systems' Multi-Jet Modeling System (MJM) that uses plastic material.

14.3.3 POWDER-BASED SYSTEMS

Powder-based manufacturing methods begin with materials that are most commonly in the form of powder, including metal, plastic, ceramic, and so on. Two representative methods and associative systems—selective laser sintering (SLS) and ZCorp's 3D Printing (3DP)—are discussed next.

The SLS process begins with slicing the STL model into its cross-sectional data. Typical slice thickness is 0.005 to 0.006 in. One powder-feed piston rises to distribute a layer of material, as illustrated in Figure 14.9(a). At the same time, the part-building cylinder lowers to the desired layer thickness. The other powder-feed piston also lowers to accommodate any surplus material, which the leveling roller transfers across the build area. Using a raster scanning pattern, the laser draws one cross-section of the desired part to sinter the powder particles. Unsintered powder remains to support the next layer. This process continues until the part is complete. The part-building cylinder is then raised to allow the part to be removed for cooling. Excess powder is cleaned off the part by brushing or air-blowing.

Building with powder materials results in rough surface finish and porosity. An advantage to this system is its ability to provide support while building. The unsintered powder surrounding the part in the build cylinder acts as a natural support for the next layer. No elaborate supports (e.g., as in some photopolymer systems) need to be built. Also, the excess powder material can be returned to the

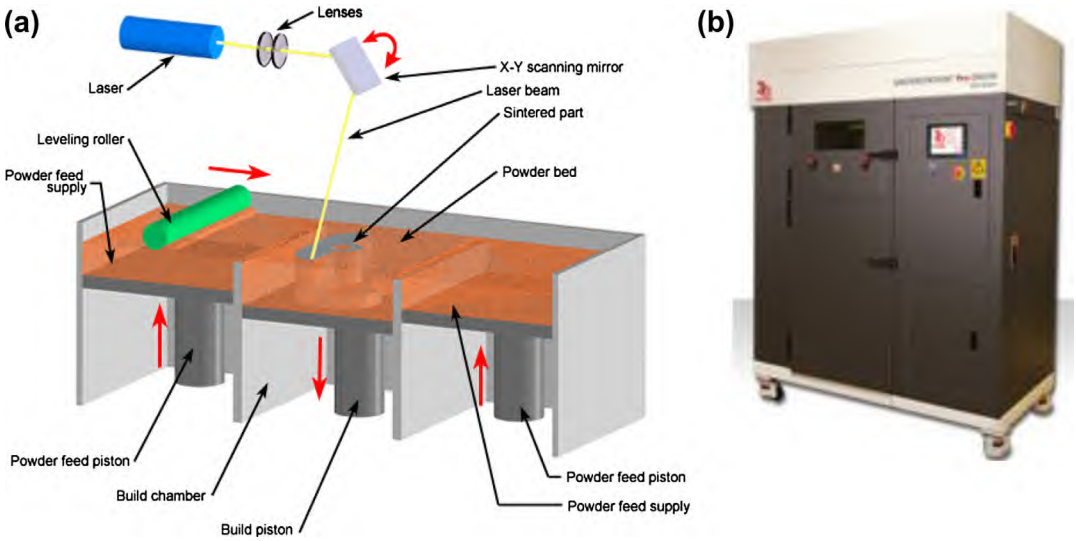


FIGURE 14.9

The selective laser sintering: (a) fabrication process (www.custompartnet.com/wu/selective-laser-sintering) and (b) the Sinterstation SLM system (www.agile-manufacturing.com/productdetails/prototyping-systems/sls-systems/sinterstation-pro-slm-systems-metals.html).

powder feed cartridges for reuse. One of the biggest problems of the sinter bonding is part shrinkage. The finished part usually has less volume than the unsintered one; therefore, void cannot be removed completely.

Several SLS systems are offered by 3D Systems, including sPro SLS series and the Sinterstation SLM (selective laser melting) system with a median price around $500K. The Sinterstation SLM system shown in Figure 14.9(b) fabricates metal parts. It can build parts as large as 9.84 in. × 9.84 in. × 13.5 in., with a typical layer thickness of 0.0008 in. to 0.004 in.

ZCorp commercializes several 3D printers, based on 3DP technology, including ZPrinter 310 Plus, ZPrinter 450, and Spectrum Z510. The 3DP technology creates 3D physical prototypes by solidifying layers of deposited powder using inkjet printing of a liquid binder. The 3DP process is shown in Figure 14.10(a). The machine spreads a layer of powder from the feedbox to cover the surface of the build piston. The printer then prints binder solution onto the loose powder, forming the first cross-section. The powder is glued together by the binder where it is printed. The remaining powder remains loose and supports the following layers that are spread and printed above it. When the cross-section is complete, the build piston is lowered, a new layer of powder is spread over its surface and the process is repeated. Once a build is complete, the excess powder is vacuumed away and the parts are lifted from the bed. One of the important advantages of 3DP is printing color models, essentially employing multiple inkjet print heads for color printing.

Powder-based RP systems of ZCorp can be among the least expensive (ranging from $20K for low-end single color to $180K for high-end full 24-bit color), although the parts tend to be weak; they must undergo a process called infiltration whereby the parts are soaked in a strengthening agent. The process

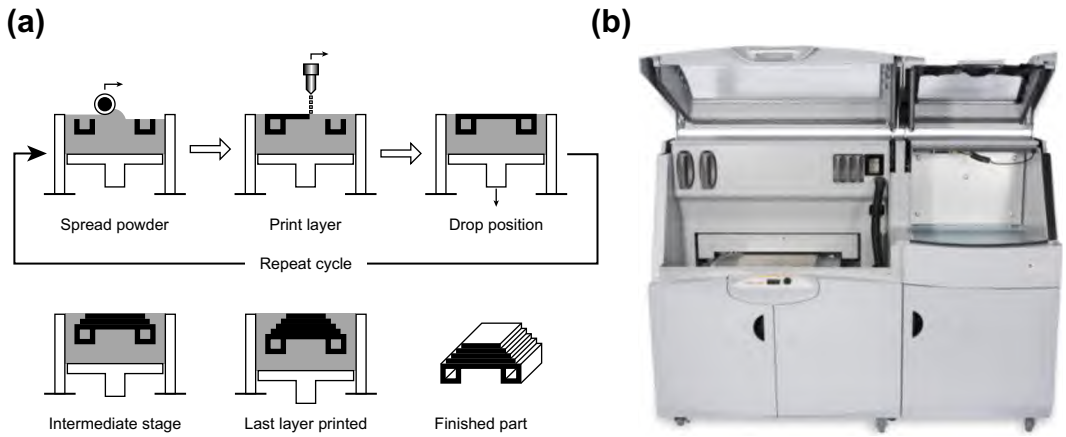


FIGURE 14.10

The 3DP technology: (a) fabrication process and (b) the ZPrinter 650 system (http://us.webkeren.org/2011/04/will-3d-printing-end-mass-manufacturing-pics.html).

does have interesting capabilities (e.g., the provision for full-color models) that could be used to show stress distributions on components or to print accurate logos or color schemes on prototypes. In addition, infiltrants can be used to give components flexible properties, allowing them to more accurately resemble rubber components. The work envelop of ZCorp systems can be up to 16 in. × 20 in. × 24 in., with a typical layer thickness of 0.004 in.

14.4 ADVANCED RP SYSTEMS

In addition to the RP systems discussed so far, there are numerous newly developed technologies and systems that aim at directly producing dense metal parts with mechanical properties comparable to those of machined or cast parts. Three such systems will be discussed, including ultrasonic consolidation of Solidica (*www.solidica.com*), electron beam melting (EBM) of Acram (*www.arcam.com*), and Laser Engineered Net Shaping (LENS) of Sandia (*www.sandia.gov*). RP also has been extended to support microfabrication for applications such as MEMS (microelectro-mechanical system). Micro RP will also be discussed.

14.4.1 SOLIDICA

Solidica, headquartered in Ann Arbor, Michigan, is an emerging company in the direct metal deposition market and provides advanced materials and solid-state fabrication equipment solutions. By combining the use of ultrasonics for layer-by-layer material buildup of metallic ribbons with a simple machining head, Solidica achieves net shape fully dense metallic components in a fraction of the time and at a lower cost than traditional machining or casting.

When bonding material through ultrasonic bonding or welding, the energy required comes in the form of mechanical vibrations. In general, the welding tool (sonotrode) couples to the part to be

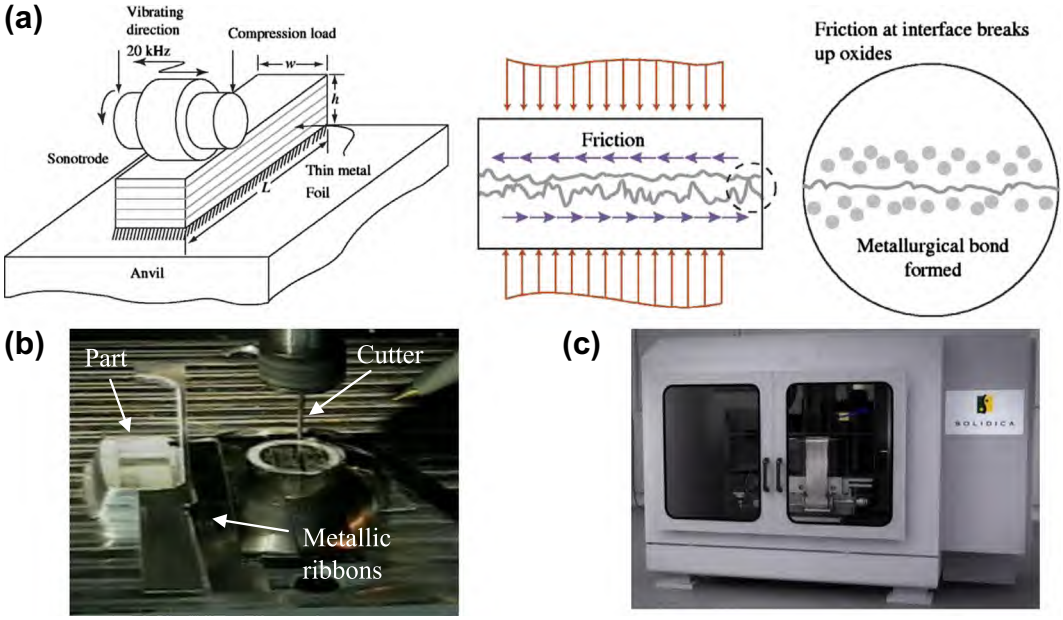


FIGURE 14.11

The RP technology of Solidica: (a) build process and ultrasonic consolidation technology (www.emeraldinsight.com/content_images/fig/1560160407035.png), (b) 3-axis CNC profile milling, and (c) the Formation 2030 system.

welded and moves it in a longitudinal direction; the part to be welded on remains static. Now the parts to be bonded are simultaneously pressed together. The simultaneous action of static and dynamic forces causes a fusion of the parts without having to use additional material, as illustrated in Figure 14.11(a).

The Solidica machine uses thin strips of metal (e.g., aluminum) as build materials. As soon as a material layer is bonded to the previous one, its exterior- and interior-boundary contours are cut to their precise geometry using a 3-axis CNC profiling milling similar to that of Figure 14.11(b). Electronic components (e.g., fiber optics or sensors) can be embedded between layers. Mixed material layers are also possible for higher strength. Formation 2030 developed by Solidica based on ultrasonic consolidation technology is shown in Figure 14.11(c).

14.4.2 ELECTRON BEAM MELTING

Electron beam melting (EBM) is a new alternative for rapid manufacturing and prototyping metal components. This technology is fast gaining attention for its ability to deliver fully dense parts, with properties equal to wrought materials, at a cost and speed substantially less than metal-based additive-fabrication methods. EBM not only creates unprecedented strength-to-weight ratios, reducing the cost

of raw materials and the weight of the component, but it also opens the door to new design configurations.

EBM technology stands out for its ability to produce titanium parts in hours versus days. For industries, such as aerospace, this technology creates new opportunities for prototyping and low-volume production of components. The time, cost, and challenges of machining or investment casting are eliminated, making titanium parts readily available for functional testing or installation on mechanical systems. EBM is patented by Arcam (*www.arcam.com*) and distributed in the United States by Stratasys.

As the name implies, EBM uses an electron beam to melt titanium powder. The additive fabrication processes build parts on a layer-by-layer basis. After melting and solidifying one layer of titanium powder, the process is repeated for subsequent layers. Within the electron beam gun, a tungsten filament incandesces and boils off a cloud of electrons (Figure 14.12(a)). These electrons stream through the gun at approximately half the speed of light. Two magnetic fields organize and direct the fast-moving electrons. The first one acts as a magnetic lens, which focuses the beam to the desired diameter. The second magnetic field deflects the focused beam to the target point on the powder bed. When the high-speed electrons strike the metal powder, the kinetic energy is instantly converted into thermal energy. Raising the temperature above the melting point, the electron beam rapidly liquefies the titanium powder. Arcam A^2 (Figure 14.12(b)) developed by Arcam is capable of fabricating parts up to 7.87 in. $\times$ 7.87 in. $\times$ 13.0 in. (200 mm $\times$ 200 mm $\times$ 330 mm).

Parts produced with EBM are near-net shape like those made with metal-casting processes. Since the electron beam fully melts the titanium, the liquefied metal conforms to the surrounding metal powder, which yields a surface finish similar to a precision sand casting; as a result, some light secondary machining or grinding may be required.

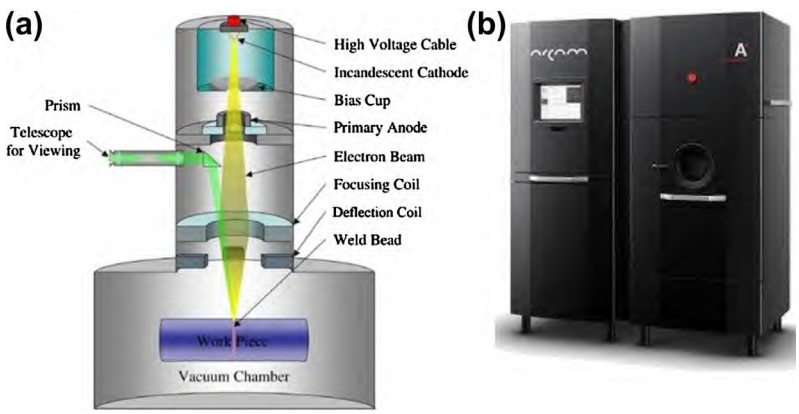


FIGURE 14.12

The EBM technology of Arcam: (a) electron beam melting diagram (www.mechanicalengineeringblog.com/wp-content/uploads/2011/03/03EBMElectronBeamMeltingrapidPrototypingWeldingworkingmodel.jpg), (b) Arcam A^2 system (http://www.arcam.com/technology/products/arcam-a2/).

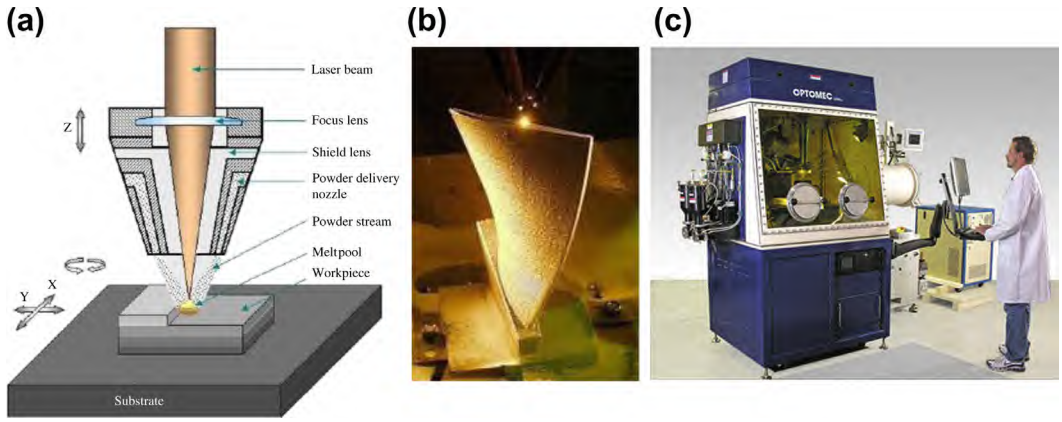


FIGURE 14.13

The LENS technology: (a) the laser engineered net shaping diagram (www.chms.ucdavis.edu/research/web/schoenung/photos/lens.jpg), (b) a closer look at the LENS fabrication (www.mechanicalengineeringblog.com/wp-content/uploads/2011/04/01laserengineeringnetshapingLENS.jpg), and (c) the Optomec LENS 850 system (www.optomec.com/Additive-Manufacturing-Systems/Overview).

14.4.3 LASER ENGINEERED NET SHAPING

Laser engineered net shaping (LENS) uses computer-controlled lasers that, in hours, weld air-blown streams of metallic powders into custom parts and manufacturing molds. The technique produces shapes close enough to the final product to eliminate the need for rough machining. One of the purposes of LENS is to make small lots of high-density parts or molds.

Nozzles each direct a stream of metal powder at a central point beneath them. Simultaneously, that point is heated by a high-powered laser beam. The laser and jets remain stationary while the model and its substrate are moved to provide new targets on which to deposit metal continually, as shown in Figure 14.13(a). First, this is done on a substrate, and then on the builtup layers, until the desired cross-sectional geometry is completed with production of a 3D metal product (Figure 14.13(b)). This is a complicated operation because high temperatures make it difficult to form accurate, smooth objects from molten metals. The technology can be used with a wide variety of metals including titanium, steels, copper, and aluminum.

The Optomec 850 shown in Figure 14.13(c) (*www.optomec.com*) is the latest of Optomec's RP machine based on LENS technology. The work envelop of 850 is 18 in. $\times$ 18 in. $\times$ 42 in. with X and Y accuracy/resolution of 0.002 in. and Z-axis accuracy/resolution of 0.020 in.

14.4.4 MICRO-MANUFACTURING RP SYSTEMS

So far we have seen a large number of different rapid prototyping technologies supporting macroscopic manufacturing. However, the need for micro parts is also largely seen in microelectronics as well as in the optoelectronics fabrication industries (Kadekar et al., 2004). Several RP technologies with potential to be used for micro or even nano fabrication processes were developed in recent years,

including micro-stereolithography (Ikuta and Maruo, 1998, 2002; Maruo and Kawata, 1998; Hanemann et al., 2006; Yang et al., 2008), micro-deposition methods (Chu et al., 2007), micro-inkjet methods (Roy, 2007), and micro-selective laser sintering (Exner et al., 2003). The beginning of using RP in microsystem manufacturing can be dated to the early 1990s (Kadekar et al., 2004), and today some of the micro-RP technologies have already been commercialized and applied to support a wide range of applications (e.g., MEMS). In this subsection, we will briefly discuss one of the representative technologies in micro-RP—the micro-stereolithography, or micro-SLA.

Micro-SLA systems can be classified into two categories—vector-by-vector micro-stereolithography and integral micro-stereolithography—based on the different laser scanning methods used. In a vector-by-vector micro-SLA system, similar to regular SLA discussed in Section 14.3.1, the laser beam is deflected by galvanometer-driven mirrors. Each layer of liquid UV resin is solidified according to the layer cross-section when being scanned by the laser spot. The micro parts are built in a point-by-point and line-by-line fashion (Hanemann et al., 2006).

In integral micro-SLA systems, each layer is built by exposure to the light source through a mask once, which significantly saves time compared to vector-by-vector systems. To achieve this, dynamic pattern generators are used to generate pattern projection for exposure curing. The time required to build a 3D structure depends only on the number of layers of the structure (Yang, 2008). In this subsection, we will briefly discuss vector-by-vector micro-stereolithography that reveals more details of the technology.

One of the vector-by-vector micro-stereolithography systems, named IH (Integrated Harden polymer stereolithography), was developed in the early 1990s, as shown schematically in Figure 14.14(b) (Maruo and Kawata, 1998). A minimum feature size of 5 μm × 5 μm × 5 μm was achieved, with the depth resolution limited to 5 μm. In this system, the light beam is no longer deflected by scanning mirrors. Instead, the focus point of the laser beam remains fixed on the surface of the resin, which produces a smaller focus spot and a higher lateral resolution, while an *X*-*Y* positioning stage moves the resin reactor in which the object is made. However, the reactor must be translated very slowly to ensure the required stability of the surface of the liquid resin.

An improved process called Mass-IH process was developed (Ikuta et al., 1996) in which, instead of an expensive UV laser system, an array of single-mode optical fibers is used to confine the optical

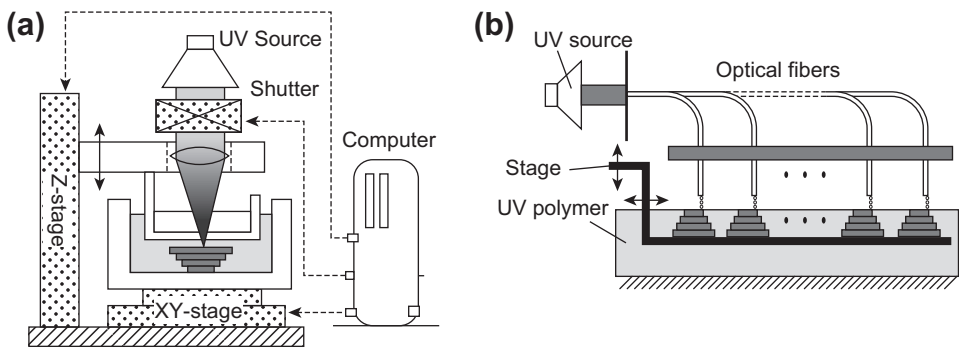


FIGURE 14.14

Micro-SLA process: (a) schematic diagram and (b) Mass-IH process (adapted from Maruo and Ikuta 1998).

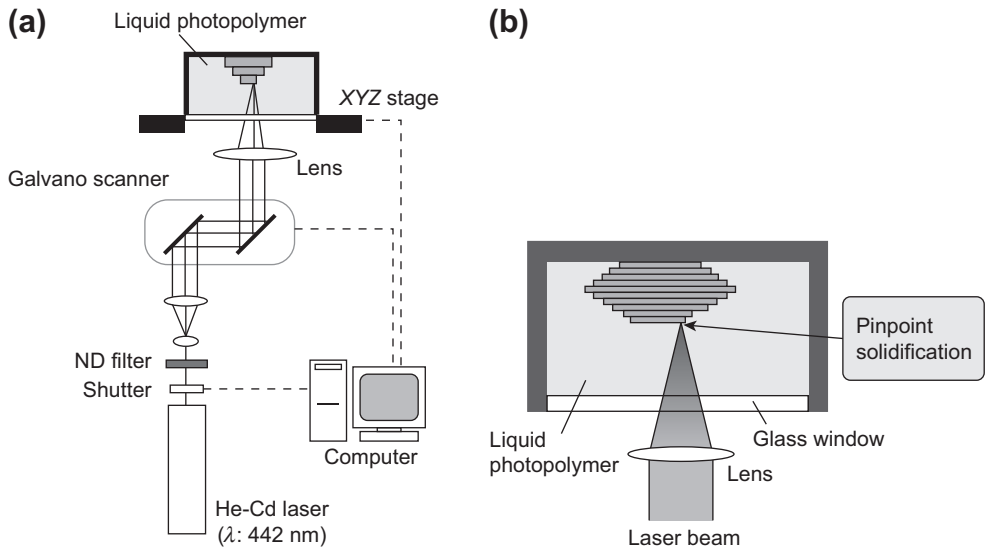


FIGURE 14.15

Super-IH micro-stereolithography: (a) fabrication system and (b) Super-IH process (adapted from Maruo and Ikuta, 2002).

path of a UV (schematic shown in Figure 14.14(b)). This method makes it possible to produce a large number of 3D microstructures at a time with several 10 μm resolution. However, the optical system for the beam delivery is complex and the substrate is also moved in the *X-Y-Z* directions in the UV-curable liquid photopolymer, which could cause the fabricated microstructures to collapse.

Then a pinpoint solidification method called Super-IH was developed during the late 1990s and early 2000s (Ikuta and Maruo, 1998; Maruo and Kawata, 1998; Maruo and Ikuta, 2002). In this process, the light source was a He–Cd laser with an oscillating wavelength of 441.6 nm and the laser beam is focused inside a commercial UV-sensitive photocurable resin that is polymerized only in a small-volume element locally; resolution was attained to less than 1 μm (Figure 14.15). By applying the Super-IH process, the resin does not need to be layered and nano-sized structures with a spatial resolution of 120 nm have been fabricated precisely. A few sample parts manufactured by the Super-IH are shown in Figure 14.16.

14.5 RAPID PROTOTYPING APPLICATIONS

As a powerful and flexible tool for product manufacturing, rapid prototyping has been used extensively in various fields in industry, providing considerable progress toward improvements. In this section, several major areas of RP application will be introduced, including design, manufacturing, art, medical science and bioengineering, personal 3D printing, and others (e.g., building construction).

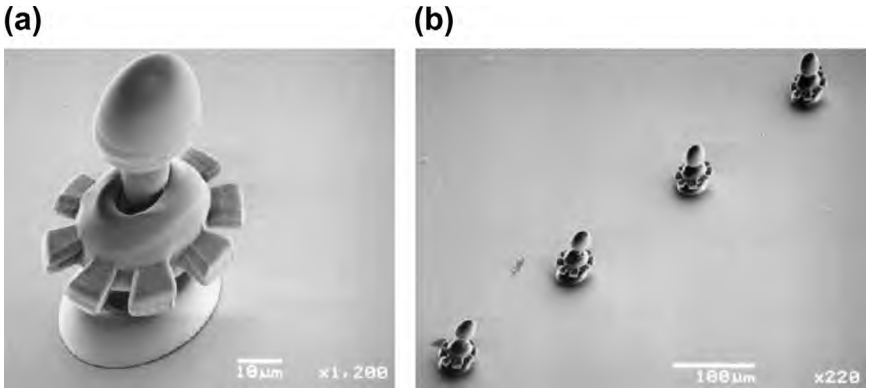


FIGURE 14.16

SEM images of movable microstructures made by the Super-IH process: (a) micro gear (diameter: 47 μm, fabrication time: 20 min) and (b) a series of micro gears (Maruo and Ikuta, 2002).

14.5.1 DESIGN APPLICATIONS

In product development, before production of a product begins, a sample or prototype is often required as part of the design cycle to allow demonstration, evaluation, verification, or testing of the proposed product. Traditionally, the prototyping process requires considerable skilled hand labor, time, and expense; typically applied to cutting, bending, shaping, and assembling a part from standard stock material.

With the help of rapid prototyping, in which a part of an arbitrary shape can be produced by adding successive layers of material, prototype models can be created fast, generally before specialized molds, tools, or jigs are designed. Designers are able to test new designs and communicate their ideas with industrial partners. The design and production cycle is shortened at the same time. Rapid prototyping has been used extensively by various industries. The Rapid Prototyping lab at General Motors employs SLA and SLS to support future products from Chevrolet, Buick, GMC, and Cadillac, especially in fashioning components, intricate subassemblies, and entire scale-model cars. One specific example is car body designed for aerodynamic performance, in which drag coefficient (CD) is to be measured and minimized. Drag coefficient is the measurement of how drag force impacts a vehicle's ability to efficiently travel through wind resistance. The smaller the CD value, the more aerodynamic the vehicle and the better the fuel efficiency of the vehicle becomes. Several major components affect CD. For example, rearview mirrors can be designed to deflect wind away from the car; headlights and front fascia can be adjusted to direct airflow smoothly along the side of the vehicle; and shutters, which close when additional engine cooling isn't required, can be installed in the lower grill.

All these designs must be tested in a wind tunnel (Figure 14.17(a)) to measure the CD. Parts that affect the aerodynamic performance of the vehicle (e.g., rearview mirror, front fascia) can be produced using RP (see example RP parts in Figure 14.17(b)); they replicate the actual surface geometry from designers, even when production tooling is not ready. These parts can be mounted on the vehicle for the wind-tunnel test. Moreover, in case redesign is necessary for improvement of the vehicle's CD, using

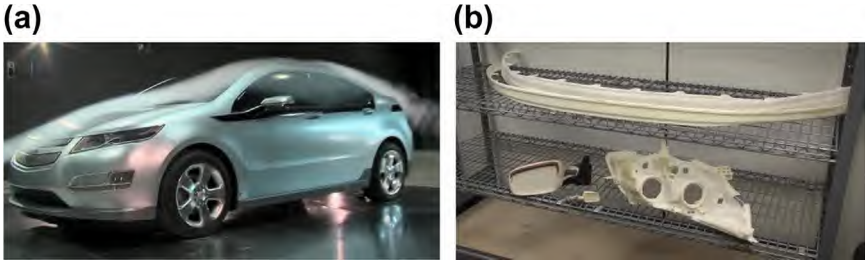


FIGURE 14.17

Example of design applications: (a) wind-tunnel test and (b) parts made by using RP for wind-tunnel test (www.youtube.com/watch?v=59xLKDfL1a8).

RP to fabricate parts to support design iterations significantly shortens development time, reduces cost, and improves product quality.

14.5.2 MANUFACTURING APPLICATIONS

Rapid prototyping technology has promise for revolutionizing many traditional manufacturing procedures such as casting. Models created by RP can be used as patterns for making molds and dies through, for example, sand casting and investment casting.

Making sand-casting patterns is costly and time consuming, especially when only a small quantity of parts is needed. In this case, we can use rapid prototyping to make a positive pattern of the certain product and then build a sand mold around it. Traditionally, wood models will be used for this purpose. As a result, the LOM (laminated object manufacturing) technology (*www.cubictechnology.com*), which uses sheet papers to build the models, is often applied to sand casting, offering an economic alternative option.

Conventional investment casting patterns are made from wax, therefore it is also called wax casting. The casting process usually requires the patterns to be dimensionally stable with temperature. To achieve this, special investment-casting wax has been invented by several RP companies (e.g., Stratasys and Cubital). Since paper material expands little with temperature rises and will burn out easily, paper LOM models are also widely used. Some other companies have developed materials and RP technologies that can be applied to investment casting. 3D Systems developed QuickCast, a build style with solid outer skin and a mostly hollow inner structure, this ensures that the part will collapse inward when heated.

Because its models can achieve multiple characteristics, rapid prototyping is being used in many other casting processes (e.g., vacuum casting and injection molding) creating positive patterns with lower cost and require less time.

14.5.3 ART APPLICATIONS

Rapid prototyping offers advantages in many fields of art. In addition to designing art works, it also contributes a lot in museum preservation and brings lifelike props to the screens.

14.5.3.1 Art Design

Just like painters are taking advantage of electronic drawing boards and printers, today, artists can also design with computers and print out their works with the help of rapid prototyping.

For sculptors, most of the forms they create are complicated, requiring much time and effort if produced completely manually. Using RP technology, the sculpting process can be finished in weeks or days rather than months or years. Sculptors would then be able to spend more time on revising their works and coming up with new ideas. They can improve their designs in the computer any number of times until the person feels the designs are perfect. Once a new conceptual idea is finalized, a sculptor can generate the physical model rapidly at any time using RP. In fact, some have already used this technology in designing and creating artworks (Séquin, 2005). A few of them have even held displays and exhibitions of their rapid sculpture projects. In the near future, everyone will be able to design furnishings or art works. All one will have to do is to create models with computers and import the digital files into RP machines. It is also possible to purchase customized designs online and print them out at home or through a service bureau. More on personal RP machines will be discussed later in this section.

In addition to art design, there are more signs that RP is relevant to fashion audiences around the world; the evidence is in recent fashion shows. For example, the 3D Fabulous Fashion Show, as part of RAPID 2012 conference in Atlanta (rapid.sme.org), featured accessories exclusively made using 3D printing technology, which can be used to quickly and quite inexpensively manufacture anything from hats to shoes (see YouTube videos: www.youtube.com/watch?v=s6uP18pFxxM and www.digitalstyledigest.com/2012/05/meetthe-worlds-first-3d-printing-fashion-shows). Another example comes in the form of the Cell Cycle Cuffs by Massachusetts-based Nervous System (see example images at www.n-e-r-v-o-u-s.com). Designed using custom software and printed on a 3D machine, the c-shaped cuffs have double-layer construction that designers say would be impossible to create using traditional manufacturing methods.

14.5.3.2 Museum Application

It has long been realized that many historical relics in museums are either degrading or have disappeared because of various reasons. Now it is possible to protect the invaluable collections using rapid prototyping technology; with it we can reproduce or create full-size or scale models of objects stored in museums.

To start the process, a 3D scanner or digitizer is needed to collect the digital data of a certain object (e.g., sculpture). Then, with the help of CAD software, a digital file can be generated according to the data imported from the scanner. During this process, the 3D model might have to be corrected to make sure that the features of the object are accurate and undistorted. Once completed, the file will be sent to a rapid prototyping machine to print the model. Many RP technologies, such as 3D Printing and SLA, have been used for museum preservation projects.

The replicas produced with RP technology are rigid and accurate, showing all the details of the original, even in very small scale. At the same time, museum collections are preserved by being converted into 3D forms on computers. With the help of printed 3D models of scanned museum collections, visitors are provided with a hands-on perspective of valuable rarities instead of them being locked away behind glass, and visitors can see more details (Materials World, 2002). In addition, scale models of historic sites and buildings can be brought into museums. Most important, the whole process can be completed without even touching the objects being reproduced.

Paleontologists at the Smithsonian Institution in Washington, DC, used rapid prototyping to make patterns for casting replacement bones of a triceratops skeleton. With 3D digital renderings of the skull, workers from the Hunterian Museum in Glasgow (*www.gla.ac.uk/hunterian*) were able to generate a physical model using SLA of the Dicynodont skull without compromising the original mold fossil. The 3D printed Dicynodont skull can be seen on display at the Elgin Museum in Scotland (*3D Printing News*, February 2012). In the future, small museums and schools will also be able to download and fabricate files to create their "instant exhibits." The replicas (e.g., skulls of primitives or scale models of dinosaur skeletons) can be presented in any classroom and passed around among students. Rapid prototyping provides a precious legacy for future generations.

14.5.3.3 Props

Rapid prototyping also plays a significant role in film production. Since 3D printing appeared in *Jurassic Park III*, the technology has come into its own in the world of special effects. It has been used in quite a number of movies to create lifelike models such as ancient creatures and magical architectural structures. During the production of *Iron Man 2*, instead of discussing conceptual ideas, the design team started its work in a 3D world. Tangible part models of the fancy armor could be available right after one design proposal was made. The entire development process was thus accelerated with the help of RP technology. Pieces for Whiplash, the film's villain played by Mickey Rourke, were printed directly for his armbands and weapons (see develop3d.com/profiles/armour-up). Also, the armor for *Iron Man 2* was designed and built based on scans taken from the actor's hand. The use of 3D printing technology allowed for objects such as Stark's Iron Man suit and Whiplash's body armor to be produced within hours, all at the touch of a button, with all that remains prior to shooting a simple painting job (3D Printing News, 2011). A 3D printer was also used to print out a Na'vi from Avatar, which supported the film staff to visualize the film's blue-skinned stars (see develop3d.com/profiles/armour-up).

Today, 3D maps are also widely used in movies and exhibitions. Map makers no longer have to struggle with ways to represent the Earth on flat paper. Rapid prototyping technology makes it possible to build multicolored physical models of three-dimensional cartographic objects according to statistical data imported. Covered by satellite imagery or topographic maps, these 3D maps (e.g., a city model) can show all the geographical features in a more direct way. For example, a 3D map representing the average prices in Euro/m^2 for building lots in Germany is shown in Figure 14.18 (Rase, 2012).

14.5.4 MEDICAL APPLICATIONS

In recent years, rapid prototyping and related technologies have brought a new dimension to the field of medicine. 3D physical models are widely used in surgical planning and the research of many medical branches, including presurgical planning and rehearsal, surgical implants, operative dentistry, and so on.

14.5.4.1 Presurgery Planning and Rehearsal

Some surgical operations, such as craniofacial and maxillofacial surgeries, are complex and require great precision and dexterity. Traditionally, surgeons discuss cases and conduct presurgical planning with the help of 2D tools such as X-ray and CT scan. Along with the development of RP technology,

FIGURE 14.18

Physical model of an interpolated smooth surface representing the average prices in Euro/m^2 for building lots in Germany (www.wdrase.de).

3D anatomical models have been introduced to help surgeons make diagnoses and suggest procedures. In 2002, doctors at UCLA's Mattel Hospital used rapid prototyping models to plan the complicated procedure of separating conjoined twins (see image of the RP models of the skulls of conjoined twins at www.turkcadcam.net/rapor/otoinsa/uyg-medikal-conjoined-twins.html). Also, to help conduct a back surgery, Sandia Lab made the accurate RP replica of a patient's spine (Figure 14.19) at the area where the surgery was to occur and solved the problem of how to maintain proper alignment of the pedicle screws (Murphy, 2002).

As a new approach for surgical planning, 3D physical models of the skull or other body part can give surgeons a realistic impression of complex structures before surgical intervention, making it easier to evaluate different approaches to surgery and to determine the best way to proceed to the

FIGURE 14.19

Rapid prototyping in medical applications: RP replica of a patient's spine (www.sandia.gov/mst/pdf/labnews05-17-02_P7.pdf).

surgical operations. The clarity of a tangible rapid prototyping model assists with the diagnosis of the condition and decreases potential miscommunication among medical professionals. On the other hand, based on spatially accurate data, these physical models can also be used as surgical navigation tools.

More importantly, rapid prototyping becomes an ideal solution for surgical simulation. For surgeons, the hands-on experience that a 3D model provides is far from matching any 2D studies. Practicing on a precise model helps surgeons educate their hands and get familiar with the progress of a complicated operation, ensuring that the operation goes exactly as planned. This "touch-to-comprehend" interaction produces more confidence for surgeons by reducing the number of unknowns during the actual operation.

A case study reported by Medtronic Australia (*www.medtronic.com.au*), called Titanium Customization for Skull Repair, represents a typical such application. The titanium patient-specific implant replaces bony voids in the cranial/craniofacial skeleton, as shown in Figure 14.20(a) (Anatomics). The implants are preshaped to fit the individual anatomy of the patient. The implants are provided nonsterile to allow preoperative use by the referring surgeon—for example, to determine the optimal fixation points. The implant can be sterilized and used intra-operatively to guide orientation. One thing worth mentioning is that stereolithography is often applied in surgical planning. The stereolithography models are transparent, providing distinct visualization of tumors or other anomalies within the surrounding tissue or bone.

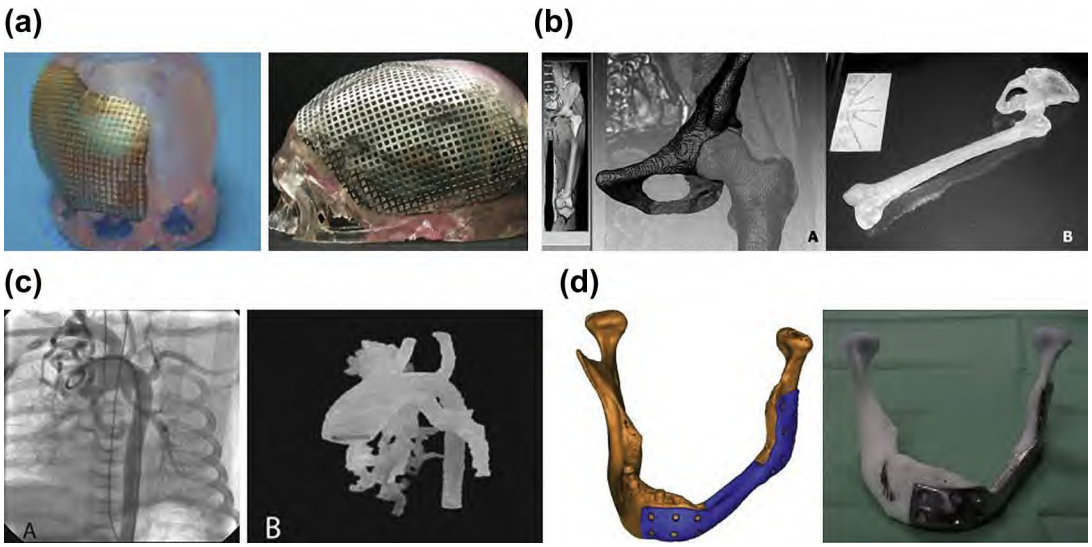


FIGURE 14.20

Examples of RP models used in teaching and surgical planning: (a) Titanium Customization for Skull Repair (www.anatomics.com); (b) THR triangulated mesh (A) and stereolithographic reconstruction (B) (Momi et al. 2005); (c) comparison of angiogram (A) with model (B) of thoracic aorta and MAPCAs (major aortopulmonary collateral arteries) in patient with pulmonary atresia (Ngan et al., 2006); and (d) jaw implant, the designed implant (left) and the milled jaw implant on the ABS model (right) (Beerens et al.—materialise.com).

Another example involves developing a tool for preoperatively planning the total hip replacement (THR), as shown in Figure 14.20(b). Starting from the MRI images, the 3D surface model of both the pelvis and the femur was built and the surgical operation was virtually performed. Data coming from gait analysis were added to visualize the physiologic movement of the hip joint. The resulting triangular mesh was sufficiently accurate to allow the building of the stereolithographic model of the joint. The plastic bones allow the user to have an enhanced vision of the surgical procedure to be performed (Momi et al., 2005).

Another such study was to assess the utility and accuracy of solid anatomic models constructed with RP technology for surgical planning in patients with pulmonary atresia with ventricular septal defect and major aortopulmonary collateral arteries (Ngan et al., 2006). The study concluded that anatomic models, such as that of Figure 14.20(c), are an intuitive means of communicating complex imaging data, such as the pulmonary vascular tree, which can be referenced intraoperatively more than is possible with conventional techniques.

One of the exciting stories involving RP for surgical planning was reported at the Materialise website (Beerens et al., *materialise.com*) in which a patient's conventional jaw implant had become infected. Starting with CT images of the patient's mandibular, a surface model was constructed of the bone defect that clearly displayed the extent of the infection. The surface model was used to design two cutting guides that fit the mandibular exactly and to design the implant. The implant design was produced through conventional milling, while the cutting guides and a 3D model of the jaw were made in ABS by fused deposition manufacturing (FDM), as shown in Figure 14.20(d). The surgical team used this model to practice the surgery. The combination of virtual planning, the ABS model, and the cutting guides substantially enhanced the surgical intervention's precision, efficiency, and simplicity. This case clearly illustrates that it is possible to use RP technology for the design and insertion of implants. Rapid prototyping technology enables faster, more accurate, and better planned implant surgeries.

14.5.4.2 Education and Research Application

Surgeons are not the only people interested in 3D physical models. RP and related technologies have enabled scientists, including anthropologists and paleontologists, to put a recognizable face on different parts of the human body and to share precise replicas of different anatomical cases. One example is a human middle ear, in which its true anatomy is revealed; the human temporal bone was captured accurately and then fabricated using SLA-7000, as shown in Figure 14.21. This scaled-up physical model ear can be used for geometric verification, education, and surgical planning and rehearsal (Sun et al., 2002). These RP models simplify so much of the communication between medical researchers, between medical professors and students, and generally between the physician and the patient.

Another such study involves manufacturing several anatomical models in a polymeric material, polydimensiloxane (PDMS) to study blood flow through a carotid artery bifurcation (Freitas et al., 2010). Based on a human carotid computerized tomography (CT) the 3D models were developed through the application of two RP techniques: fused deposition modeling (FDM) and 3D printing (3DP). By combining the rapid prototyping techniques with a PDMS casting technique, it was possible to obtain an anatomically transparent model of a human carotid artery made by an elastomeric material (i.e., PDMS). This combination is a promising technique to perform for in vitro blood studies through anatomically realistic models such as a carotid artery.

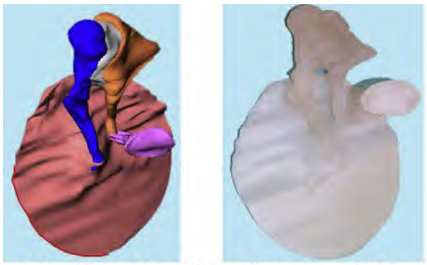


FIGURE 14.21

Examples of RP for medical research: human middle ear model using SLA—CAD solid model on the left and physical replica of SLA model on the right (Sun et al., 2002).

14.5.4.3 Dental Applications

Since a dental model can be very difficult to fabricate using the conventional subtracting method because of its complexity, rapid prototyping is becoming more attractive in dental applications. In orthodontics, RP models can be used to design orthodontic treatment devices with the specific patient's tooth alignment. By scanning a patient's teeth using a CT/MRI scanner or laser digitizer, a model of the tooth's condition can be precisely built, including specific tooth alignment characteristics. It allows for development of an orthotic device that improves the fit, comfort, and stability. One such example involves metal casting for a molar crown (Liu et al., 2005). The original tooth surface of the incisor is CT scanned and saved as the outer crown surface data. Then the outer surface of the incisor is ground to simulate the cleaning operation of caries and the ground surface is scanned and saved as the inner surface data. Pro/ENGINEER software is used to construct the 3D model. Model Maker is used to make a wax pattern from the 3D model. The metallic crown fabricated by investment casting from the RP crown model is similar to the crown fabricated by the traditional process, as shown in Figure 14.22.

Another example from Chang et al. (2003) involves constructing accurate computer and physical models that can be employed for the study of human tooth mechanics using images scanned from a

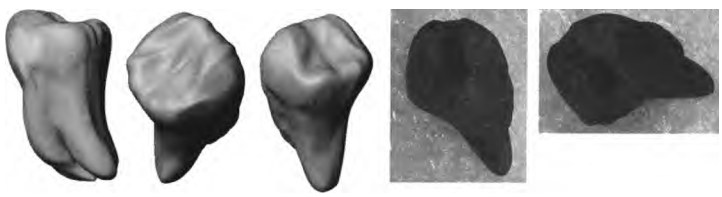


FIGURE 14.22

RP for dental applications: CAD model and wax model of human first molar fabricated using Model Maker II (Chang et al., 2003).

histological section preparation of a human tooth (Figure 14.22). RP technology is also helpful for dental surgery planning and diagnosis, and it can be used in the manufacture of dental devices (e.g., drill guides). Since holes have to be drilled in the patient's jawbone in order for the dental implants to be placed, the design of the drill guides requires accurate location and orientation; this can be achieved easily using RP devices. One such example is a custom plastic dental drilling guide made on a 3D Systems SLA machine (Waterman, 2010). This aids dentists in drilling holes for inserting tooth implants.

In addition, RP dental models can be very beneficial to the anthropologist. In cases where only one or two specimens exist, rapid prototyping allows for replication of jawbone and teeth so that molding, measuring, and dissecting of the remains can be done without causing harm. Such models can also be used to show changes in evolution (Liu et al., 2005).

14.5.5 BIOENGINEERING APPLICATIONS

Rapid prototyping provides new possibilities for custom prosthesis and implantation. Moreover, it becomes a better approach for fabricating tissue engineering scaffolds.

14.5.5.1 Custom Prosthesis and Implantation

Rapid prototyping has brought great improvements to the fields of prosthetics and implantation. As there are always patients outside the standard range—between sizes—or with special requirements caused by disease or genetics, it is of great importance to manufacture a customized prosthesis that precisely fits a patient. This is because in hip, knee, shoulder, or any joint replacement, it is crucial that there is secure fixation so that the implant does not move. If there is too much relative movement between an implant and a bone, weak tissue will grow around the implant instead of hard bone tissue. In addition, loosely fitting implants ultimately cause pain, reduce function, and often have to be removed and replaced in another operation.

One example is an OsteoAnchor (*www.osteoanchor.com*) hip replacement in which a rapid prototyping technology called DMLS (direct metal laser sintering)—essentially SLS was employed (Siliconrepublic, 2012). The DMLS starts off with a bed of very fine, loose titanium powder. A laser on the top of the machine points at the bed and sinters out a 2D slice of implant before more loose powder is spread on top. The laser then sinters the next layer to the previous one and so on, thereby building the implant slice by slice. The preclinical study of an OsteoAnchor hip replacement showed that the technology immediately grips the bone effectively (see article at www.siliconrepublic.com/innovation/item/30399-medtech); recovery after surgery is quicker; and, most essentially, the hip replacements remain very secure as a result of extensive bone in-growth.

Sometimes implants are made directly with rapid prototyping. Scientists at Manchester University have developed a RP printer that is able to print a tailor-made strip of human skin (see article at www.manchestereveningnews.co.uk/news/greater-manchester-news/tailor-made-skin-from-ink-printer-1054590) that can be sewn onto the body (Camber, 2005). The cells are put into a special printer ink liquid and artificially multiplied. Then, the printer prints the cells onto a plastic surface, which acts like a scaffold to support the cells. The plastic could then be surgically attached to the damaged part of the body and the plastic would dissolve naturally, allowing the body to use the strip of cells to repair the injury.

A design process was recently proposed by Parthasarathy et al. (2011) for creating periodic cellular structures specifically targeted for biomedical applications. Electron beam melting is used to fabricate the parts. Implants designed and fabricated with this design strategy and manufacturing process would have mechanical properties equivalent to the part they replace and restore better function and esthetics as opposed to the currently used methods of reconstruction. An example of a titanium porous cranioplasty plate fabricated using this proposed method is shown in Figure 14.23.

14.5.5.2 Scaffolds for Tissue Engineering

Tissue engineering typically involves the assembly of tissue structures by combining cells and biomaterials with the ultimate goal of replacing, or restoring, physiological functions lost in diseased or damaged organs. Stem cells are able to express all the necessary proteins to create a complex organ (e.g., a heart or a liver); this is, if they are held in the correct geometric format while the structure's materials must "melt away" or metabolize in some way as the organ develops. The biomaterial scaffolds are thus designed to provide mechanical support for the cells. However, more complex

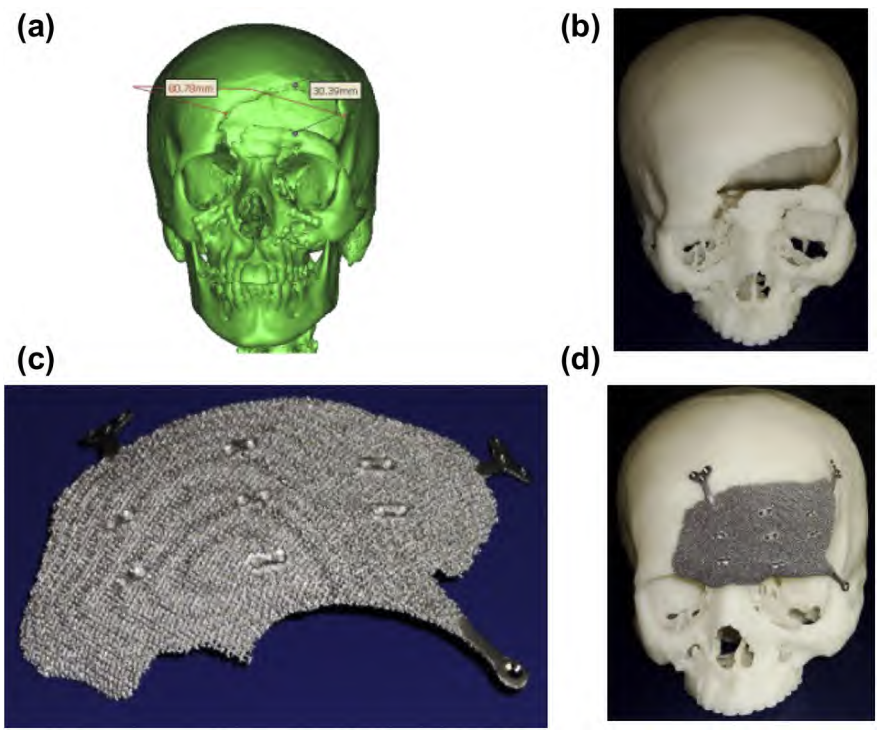


FIGURE 14.23

RP for periodic cellular structures: (a) 3D digital reconstruction of skull defect, (b) skull model, (c) porous titanium implant fabricated with EBM, and (d) titanium implant fitted to the skull model (Parthasarathy et al., 2011).

architectures that mimic tissue structures have been more difficult to produce with traditional methods of production. In recent years, researchers have turned to using rapid prototyping technology in the fabrication of tissue engineering scaffolds.

With the help of rapid prototyping, it becomes possible to fabricate the multicellular structures required for complex tissue function. In addition, fabrication of vascular beds would allow the construction of larger tissue constructs. Also, custom-made scaffolds can be produced directly from clinical data on a computer. RP models have also proved useful for studying structural and functional relationships in model tissues.

While the goal of manufacturing complex tissues (e.g., liver or kidney) remains a lofty goal, RP has been used to build scaffolds for bone tissue engineering, which holds great promise for the future treatment of osseous defects (Matthias, 2006). Using RP methods, scaffolds can be generated with a predefined, well-controlled internal and external architecture that mimics the structure of natural bone. Scaffolds generated by RP techniques can be used for seeding patient-derived cells onto scaffolds serving as 3D templates for initial cell attachment and subsequent tissue formation. The ultimate goal is to produce biocompatible scaffolds with fully interconnected channels for the replacement of lost bone. Figure 14.24 illustrates a general process of porous scaffold design for tissue engineering (Hollister, 2005).

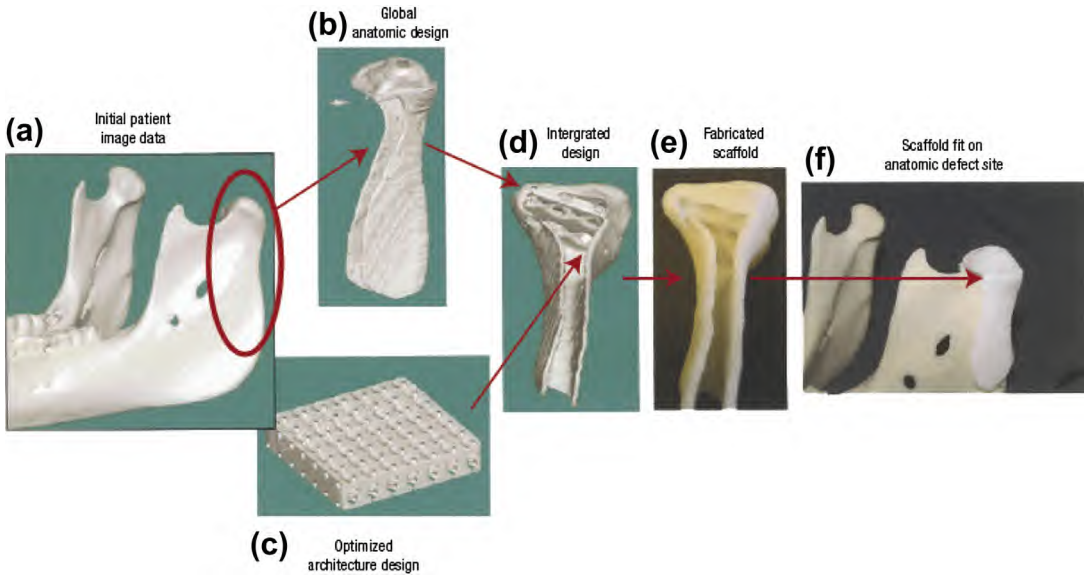


FIGURE 14.24

Image-based procedure for integrating a designed microstructure with an anatomic shape: (a) CT or MRI scan serves as starting point for designing scaffold exterior; (b) scaffold exterior shape is created with additional features for surgical fixation; (c) architecture image design is created; (d) global anatomic and architecture design are integrated using Boolean image techniques; (e) SFF is used to fabricate design from degradable biomaterial, in this case SLS was used to fabricate a scaffold; and (f) final fabricated scaffold fits well on the intended anatomic reconstruction site (Hollister, 2005).

14.5.5.3 Organ Printing

Growing new organs from scratch has already become reality. In addition to bladders (see article at www.thedailybeast.com/newsweek/2010/12/07/future-of-medicine-growing-new-organs.html), scientists have engineered new skin, bone, cartilage, corneas, windpipes, arteries, and urethras. Human organs fail for a multitude of reasons: genetic deformities, injuries, and disease can all damage them. Organ transplantation is an option, of course, but it is risky, and too often there aren't enough donated organs to meet the growing need (Komaroff, 2010).

One of the emerging research fields that involves RP for growing organs is organ printing (Visconti, 2010), also called bioprinting (see youTube video: www.youtube.com/watch?v=80DhBLEhdzk&feature=related). The "ink" in the bioprinting process used by, for example, Organovo, Inc. (*www.organovo.com*), is composed of spheres packed with tens of thousands of human cells. These spheres are assembled or printed on sheets of organic biopaper. By precisely placing the cells with the bioprinter, and providing them with the proper natural developmental cues, they do exactly what they do in nature—the cells self-assemble into fully formed, functional tissue. This unique science blends biophysics and cell biology with computer-aided design and high-precision deposition to recreate the micro-architecture of the most complex human tissue. Transplant organ printing makes it possible for new tissue to replace diseased tissue. Since new tissue can be developed from one's own body cell sources, rejection of transplanted tissue is in general not an issue. The cells can be taken from youthful progenitor cells in bone marrow to generate replacements for the older diseased cells. The cells' ability to self-assemble means they will organize themselves into a functional tissue after being positioned.

14.5.6 PERSONAL RP

After so much discussion about rapid prototyping and what the machines can do in support of a broad range of engineering and medical applications, have you ever thought of owning an RP for your personal use? RP is on the cusp of moving into businesses or living rooms and is becoming incredibly useful. Have you ever imaged scanning a crescent wrench (maybe download one from the Internet), taking the image of it, and producing a working duplicate using a 3D printer. You may even make the wrench larger, smaller, or have a bigger handle just by adjusting it on a computer, then print the unique result to solve a particular problem. In that case, you wouldn't need to buy a tool set, just print the tools you need as you need them. This will not be in your imagination any more. Several personal RP machines are available on the market, such as the Cube 3D Printer (Figure 14.25); the 3DTouch 3D Printer from 3D Systems (see images at www.3dsystems.com); and the Sintermask-fabbster-3D-printer (see image at www.flickr.com/photos/creative_tools/sets/72157631848246931). These personal RP machines are usually priced at less than $3000. Although not all personal RPs employ the same build technique, the RP machines shown in Figure 14.25 employ fused-deposition manufacturing (FDM) and strong and durable materials (e.g., ABS) to build parts.

14.5.7 OTHER APPLICATIONS

The application of rapid prototyping technology touches on many other aspects of our lives. Today, RP can be used to assist with criminal investigations in which rapid prototyping was used to build the

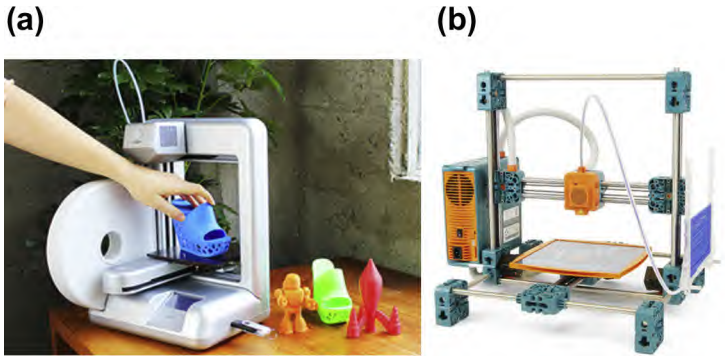


FIGURE 14.25

Personal RP machines: (a) Cube 3D Printer (cubify.com/cube), (b) Sintermask–fabbster–3D-printer (www.flickr.com/photos/creative_tools/sets/72157631848246931).

FIGURE 14.26

Some other RP applications, anthropological or craniometrical landmarks on a skull (*left*) and a reconstructed facial (*right*) (www.uta.edu/publications/utamagazine/fall_2007/index.php?id=529).

replica of the victim's skull, as shown in Figure 14.26 (Asiabanpour et al., 2008). Also, RP models are sometimes kept as evidence; they often are accurate enough to show all the details such as force effects. In court processing, tangible RP models can be presented to help recreate the crime scene (Crockett and Zick, 2000).

Rapid prototyping technologies have enabled mathematicians to put on the face of geometric models. Many mathematics objects are far too complex to be properly understood through pictures, adding difficulty to mathematics research and teaching. Now these structures (e.g., complicated polyhedrons) can be easily created with the help of a RP machine, and people will have a chance to find

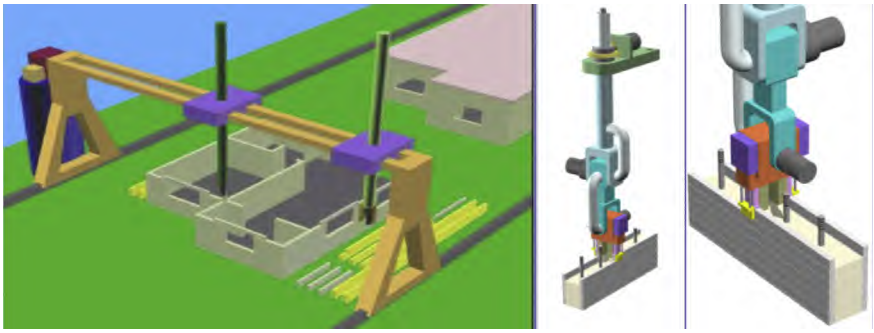


FIGURE 14.27

Contour crafting developed at the University of Southern California (www.contourcrafting.org).

the beauty of those abstract mathematical forms. One such example (see image at www.georgehart.com/rp/chiral-2-layer-sphere.html) is a 4-inch RP model of a two-layer geodesic sphere, the world's only chiral two-layer geodesic sphere. There are 260 triangles in the outer layer; the inner layer has 12 pentagons and 120 hexagons (Hart, 2005).

Some other engineers are trying to use rapid prototyping to make freeform architectural structures, so-called contour crafting (Figure 14.27). A team at the University of Southern California is working to create a robotic "house printer" that can custom-specify cement structures that may be difficult to build in other ways (*www.contourcrafting.org*).

14.6 CASE STUDY: RP FOR COMPLEX ASSEMBLY

In this section, two case studies will be presented: a single-piston engine and a Formula SAE racecar (Chang and Long, 2011). The purpose of presenting case studies is to offer more details in using RP for practical applications. Both examples involve fabricating an assembly instead of a single component. For the engine example, one extra requirement is to allow parts to move in order to show the relative motion inside the engine. For the racecar example, the challenge is dealing with a large-scale, complex assembly with many small parts.

14.6.1 SINGLE-PISTON ENGINE

A single-piston engine assembly shown in Chapter 8, Figure 8.2 was printed using Dimension 1200sst of Stratasys. A primary focus is with regard to the provision for the relative motion between the components. Also of note is that the intent is for the internal components to be printed in-place, thus requiring no assembly after it has been printed. This means that the individual parts cannot be disassembled.

The CAD geometry was created in *SolidWorks*. The engine example consists of four major components: case, propeller, connecting rod, and piston. The overall size of the engine is 6.68 in. × 8.98 in. × 8.02 in. and was printed at full scale to the CAD model. The STL was generated using

default settings in *SolidWorks*, which entailed a chord height, or deviation, of 0.00809 in., resulting in an STL file with 51,942 facets.

To prepare the model for printing, the locations where clearance for moving parts would be necessary were identified and the mate constraints were modified to allow for 0.02 in. of clearance between the surfaces of moving parts. Note that the required clearance is RP machine-dependent. In general, an excessive clearance leads to an assembly with loosely mated parts. On the other hand, an insufficient clearance may not provide adequate space for the build and removal of support material between parts, preventing relative motion between parts. The 0.02 in. clearance was obtained after a few trials.

The interference-checking capability in *SolidWorks* was used to verify that there was no interference that would cause problems for the moving parts. The model was then exported to an STL file, sliced, and printed on the Dimension machine. Figure 14.28(a) shows that the orientation of the engine model in the build chamber in *CatalystEX* software comes with Dimension 1200sst. Figure 14.28(b) shows the printed prototype and Figure 14.28(c) shows the prototype after removal of support materials. The model took 33.5 cubic in. of model material and 20.2 cubic in. of support material. It was printed using the smallest layer thickness, 0.01 in., and with the "sparse" support material option offered by the system; it took 50 hours to complete printing.

One of the challenges with this model was getting the support material removed from inside the engine's cylinder. The very small space made it difficult to break away the support material. In this case, the printed model was soaked in WaveWash, a support material cleaning system offered by Stratasys. After the support material was removed, it was found that the cylindrical shapes that were oriented in the vertical direction had much better accuracy and smoothness for the allowance of motion compared to the surfaces oriented in a horizontal direction. The reason for this is the result of the rougher surface present on the horizontally printed curves and the thickness of the individual layers of material.

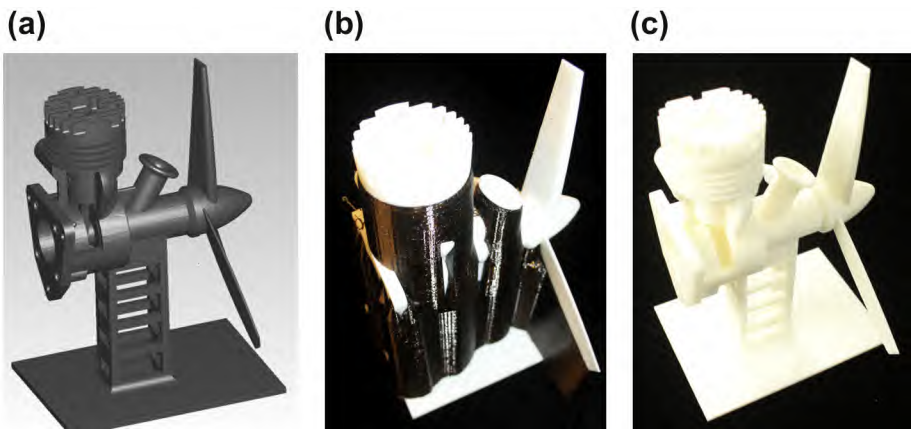


FIGURE 14.28

Physical prototype of the single-piston engine model: (a) original STL, (b) physical prototype with support material (black), and (c) physical prototype with support removed (Chang and Long, 2011).

14.6.2 FORMULA SAE RACECAR

The racecar model shown in Chapter 8, Figure 8.19 is employed to demonstrate the feasibility of using RP for fabricating large-scale and complex assembly. This is a Formula SAE-style—Society of Automotive Engineers—racecar designed and built by engineering students.

The racecar's detailed CAD model shown in Chapter 8, Figure 8.19(b) was created in Pro/ENGINEER, consisting of about 1400 distinguished parts; many are under ¼ in. diameter or thickness. The racecar was designed and later built in its full size (see Figure 8.19(a)) for the annual Formula SAE competition. The wheelbase of the car is 68 in., and the front and rear tracks are 49 in. and 48 in., respectively. The CAD model was created in such detail that it was within 0.7 lb. of the as-built car, which weights 445 lb.

14.6.2.1 Scale Factor

The first step in creating a scale model of a large assembly is to determine the scale factor. Scale factor is the primary parameter; it significantly affects the fidelity, quality, strength, and cost of the physical prototypes to be fabricated. The first and the most important consideration in determining the scale factor is the model fidelity; that is, the level of detail to be retained in the prototype. The level of fidelity is application-dependent, which will have to be determined by the designers who are considering the intended usage of the prototypes. For example, critical parts and features, such as suspension components of the racecar shown in Figure 8.21, must be identified and the size of the smallest features, or parts, must be acquired. Parts that are decorative and do not contribute to enhancing the level of fidelity (e.g., odometer of the instrumentation panel shown in Figure 14.29(a)), can be ignored. Small parts that will cause problems in fabrication, such as fuel line (Figure 14.29(c)), accelerator cable (Figure 14.29(b)), and wire harness (Figure 14.29(d)), must be also identified. Parts like these can be completely ignored while determining the scale factor.

The second aspect in determining a scale factor is the capabilities of the RP machines to be employed. One important factor is machine resolution, which, in theory, determines the minimum feature sizes that can be fabricated. However, in practice, the smallest features to be fabricated are much greater than the machine resolution, considering the strength of the parts fabricated and steps involved in postprocessing. RP machines that require building support to fabricate overhangs (e.g., Dimension 1200sst) could cause significant damage in small parts while removing the support. Similar issues may appear when retrieving a prototype from a powder bin using, for instance, ZCorp RP machines (*www.zcorp.com/en/Products/3D-Printers/spage.aspx*).

Once the scale factor is determined, small parts that fall below the threshold of the viable physical size will have to be removed from the assembly. From examining several prototypes fabricated by the Dimension printer, it was found that for a reasonable level of part strength, a thickness of 0.06 in. was necessary as a general rule of thumb. Features smaller than this tended to be extremely weak and fragile. The scale factor s can be calculated by

$$s = A/a \tag{14.1}$$

where A is the size of the smallest features in the assembly to retain, and a is the size of the smallest feature that can be fabricated with adequate strength using an RP machine. After the scale factor is determined, the envelop of the actual prototype size $\ell \times w \times d$ can be calculated by

$$\ell = L/a, \quad w = W/a, \quad \text{and} \quad d = D/a \tag{14.2}$$

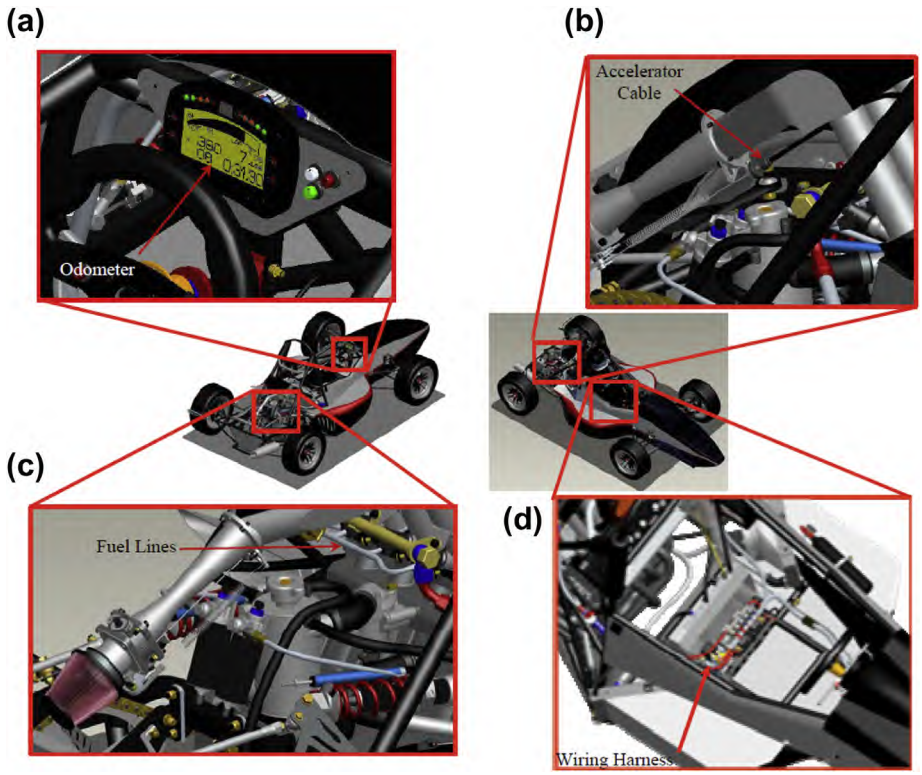


FIGURE 14.29

Small components to remove: (a) odometer, (b) accelerator cable, (c) fuel lines, and (d) wiring harness (Chang and Long, 2011).

where L, W, and D are the length, width, and depth of the envelop of the original assembly, respectively.

Note that if prototype size is too large to handle, a different RP machine may be considered for increasing the scale factor. For example, an RP machine using SLS (e.g., Sinterstation of 3D Systems) produces stronger parts; therefore, it may allow for smaller features to be built. Another possibility is to lower the level of fidelity by eliminating more parts and features, which is, however, less desirable in general. If the physical model size exceeds the build envelop of the RP machine, the model can be split into pieces. They can be built separately and physically assembled afterward. In this case, interlocking features must be added to the CAD model before fabrication to assist in aligning and joining the separate pieces of the prototype.

Adhesives are made for use on virtually every material that can be used to permanently join pieces together, such as common glues for plastics and metals. More specifically, SLA parts are joined by dispensing UVBOND SL-Lo (*www.apteklabs.com*) along the length of the bond line and applying a UV light source to cure adhesive. Also, Pro-Weld plastic bonding agent and sealant from Ambroid (*www.ambroid.com*) can be applied to bond models fabricated using a Dimension RP machine. In

general, SLS prototypes tend to be better suited for bonding; due to the porosity of an SLS part, the adhesive penetrates the part surface to form a bond that is stronger than that found when joining SLA sections.

For the racecar, the smallest features to be retained in the model were identified by measuring a number of the smaller details (e.g., body panels) of the model—most of which measured about 0.375 in. in the full-size CAD model. To identify the scale factor to be used, this number was divided by 0.06 in., which corresponds to the minimum feature size the RP machine can create with reasonable strength. This yields a scale factor of approximately 6 to 1. Since the car has a length of 115 in., the car's prototype length will be approximately 19.2 in. Given that the work envelop of the Dimension 1200sst measures 10 in. × 10 in. × 12 in., the model was split in half, yielding two halves that measure approximately 9.6 in. in length.

14.6.2.2 Model Modification

Based on the scale factor, a number of components were either modified or removed completely to improve the structural integrity of the model. Parts that are small and do not contribute to the fidelity of the physical model were removed, including fuel lines, wiring harness, and accelerator cable similar to those shown in Figure 14.29. In many cases, these components could be removed without causing constraint failures in the CAD assembly model. In instances where constraint failures were unavoidable, the problematic constraints were redefined to alternate geometry that would be persistent in the model. For example, if a brake rotor required a constraint to a brake line, the constraint would be modified so that the rotor would be constrained to the steering knuckle, or another larger component that will be retained in the printed model. Parent–child relationships in CAD must be examined carefully in advance to minimize such problems.

Parts were also modified, including increasing thickness in shells and panels, filling cavities or holes, and altering part dimensions to increase their strength in the physical model. The main body components (e.g, the car's nose cone, side pods, as well as the under tray and cooling ducts) are very thin—approximately 0.0625 in. thick in the CAD model (Figure 14.30(a)), which were also thickened. Six major panels were thickened to about 0.75 in. in the CAD model, which resulted in a wall thickness of approximately 0.125 in. on the prototype. Also, the cavities inside the brake fluid reservoirs, shown in Figure 14.30(b), are filled. In addition, the sizes of the spokes on the wheel centers (Figure 14.30(c)), as well as the sprocket (Figure 14.30), were increased such that the printed model's minimum cross-sectional thickness was greater than 0.06 in. Also, the diameter of the damper (Figure 14.30(e)) was increased to the diameter of the coil spring, minus half the diameter of the wire in the spring to provide more support for the coil. In addition, the gap between the shock absorber and the linear displacement sensor, which is depicted as a small cylinder above the coil spring in Figure 14.30(e), was filled with a thin extrusion measuring 0.5 in. thick. The 28 tubes of the frame shown in Figure 14.30(f) were completely solidified in the model, since the tubing thickness was far too thin for the 3D printer to create. The tubes along the bottom of the frame, located under the driver, could not be completely solidified because the center of the holes was referenced extensively by other geometry. To quickly get around this, the inside diameter of the tubes was minimized to 1/8 in. in CAD. This does not affect the fidelity of the physical model, yet adds strength to it.

Another type of modification involves cutting out some parts, or portion of the assembly, to allow the internal structure to become more visible. For example, a front left quarter of the body panel in the

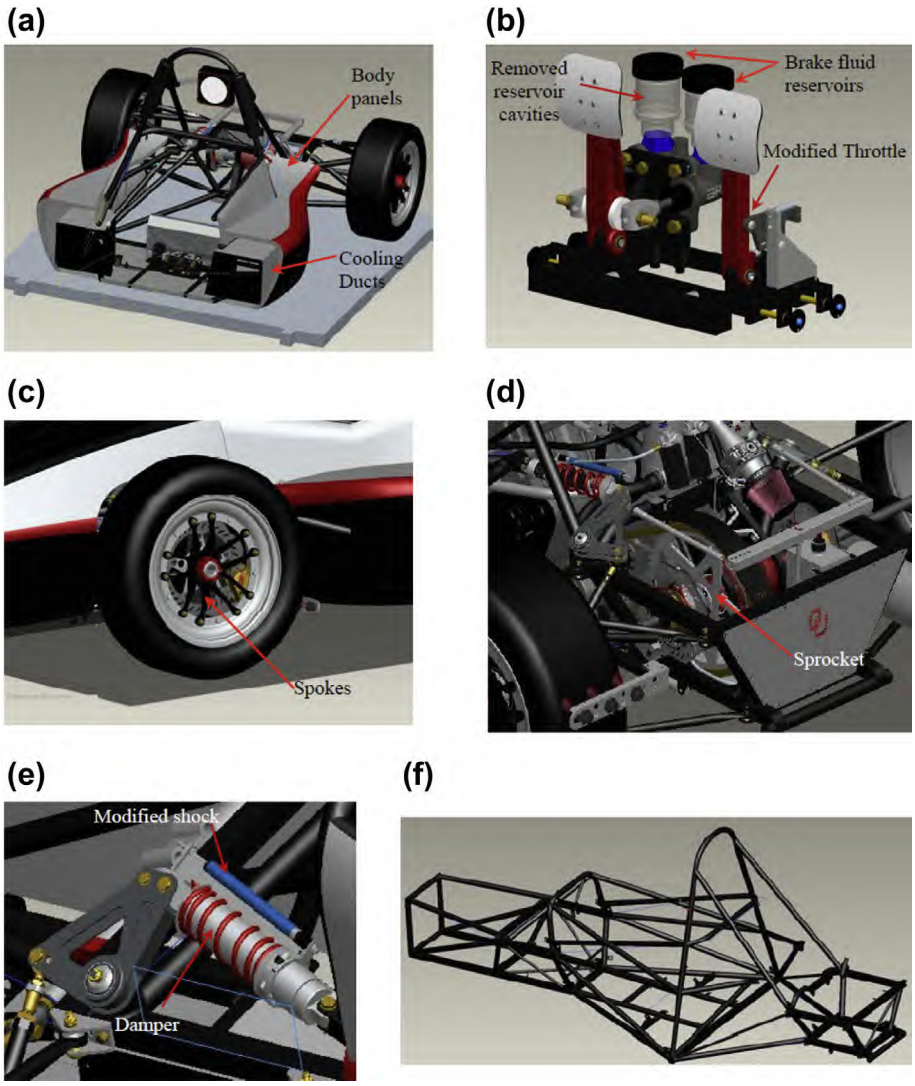


FIGURE 14.30

Parts and assemblies removed or modified: (a) body panels and thin shell parts thickened, (b) brake system, (c) spokes of the wheel center, (d) sprocket of the transmission, (e) shock and damper, and (f) solidified frame tubes (Chang and Long, 2011).

racecar is cut out to reveal the design of the brake system and steering column, and so on, as shown in Figure 14.31.

The CAD models of the front half before and after modification are shown in Figures 14.32(a) and (b), respectively. Some of the major differences include the frame tubes that are solidified; cooling

FIGURE 14.31

Front cover partially removed on racecar to reveal internal structure (Chang and Long, 2011).

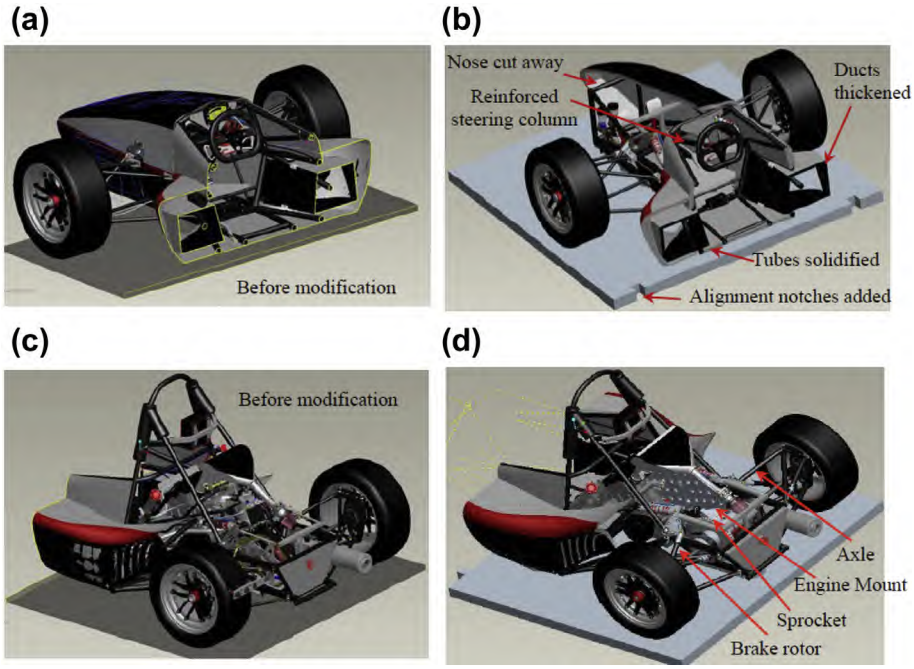


FIGURE 14.32

CAD models of the racecar before and after modifications: (a) front half before modification, (b) front half after modification, (c) rear half before modification, and (d) rear half after modification (Chang and Long, 2011).

ducts and body panels that have been thickened, or partially removed for visibility; the reinforced steering column and steering wheel; and, finally, the extruded base the model sits on. In addition, alignment notches were added, as shown in Figure 14.32(b).

The rear half of the car was treated in much the same manner. The rear shocks were modified in the same manner as the front ones were; all brake lines, cables, and hoses were removed from the assembly. Very small components (e.g., the throttle return spring and switches on the dash panel) also were removed. The thickness of many of the drive-train components was increased in order to strengthen them. Other modified parts included the rear sprocket, brake rotors, engine mounts, and axles. These components had their smaller cross-sectional dimensions increased such that they would have a printed model size of cross-sectional dimensions greater than 0.06 in. The CAD models of the rear half before and after modifications are shown in Figures 14.32(c) and (d), respectively.

Note that while altering such parts, potential model interference could occur. For example, the accelerator pedal was very weak; to improve its strength the bracket to which the pedal was pinned was altered. The space between the tabs of the bracket was filled so that the pedal in the model would physically interfere with the bracket, increasing the component's structural support while having a minimal impact on the overall detail of the component. In addition, a base was created in the model to provide a rigid platform for the delicate geometry of the car to sit on. To improve the rigidity of the model, five small support columns measuring 0.375 in. in diameter were added underneath the floor pan.

14.6.2.3 Model Conversion

The next step is to convert the Pro/ENGINEER models of the modified halves to STL for printing. There are two possible options: direct conversion to STL from CAD and conversion via the VRML (Virtual Reality Modeling Language, *en.wikipedia.org/wiki/VRML*).

The front half of the car was converted first to VRML and then to an STL model. This process was successful and repeatable with zero tessellation errors encountered. This method is a useful alternative if the available computer hardware directly creating the STL was not possible. Due to the large file size and RAM limitations, Pro/ENGINEER would crash during the conversion process to the STL file type. While using the VRML as an intermediate file format provided a successful method of creating the STL file, STL file's accuracy to the original model suffered since the model was tessellated twice.

The rear half of the car was converted to an STL from Pro/ENGINEER directly. The back half features a smoother overall appearance than the front as a result of the direct STL conversion. The differences are most notable on the rounded surfaces of the wheels, and body panels as can be seen when comparing Figures 14.33(a) and (b), as previously mentioned.

14.6.2.4 Model Fabrication

The STL was brought into *CatalystEX*, the printing software used by the Dimension 1200sst. The STL model of each half was scaled down to 1/6 to fit into the work envelop.

Once the part was oriented, the print settings were adjusted. The Support Fill option was set to *Sparse*, Layer Resolution to the finest, 0.01 in.; and the Model Interior option to *Solid-Normal* for the front half of the car. The Support Fill option for the back half of the car was set to *Minimal*. Using this setting generates simpler toolpaths for the support material and also uses a build pattern that is much easier to remove; although, the build pattern does require the use of more support material. For the

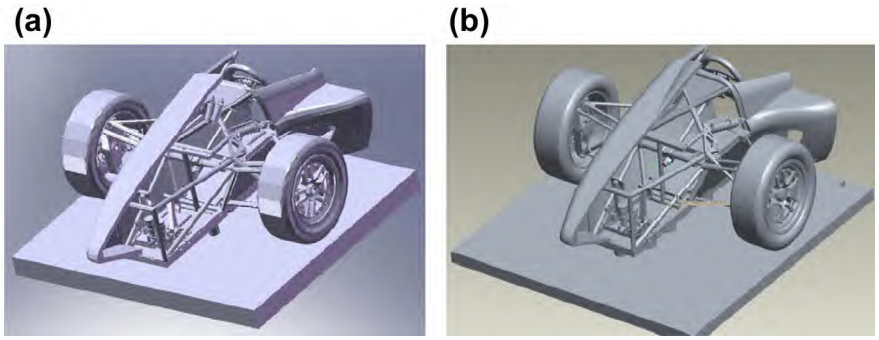


FIGURE 14.33

STL models from two methods: (a) STL created by saving as VRML first and (b) STL created by directly saving as an STL (Chang and Long, 2011).

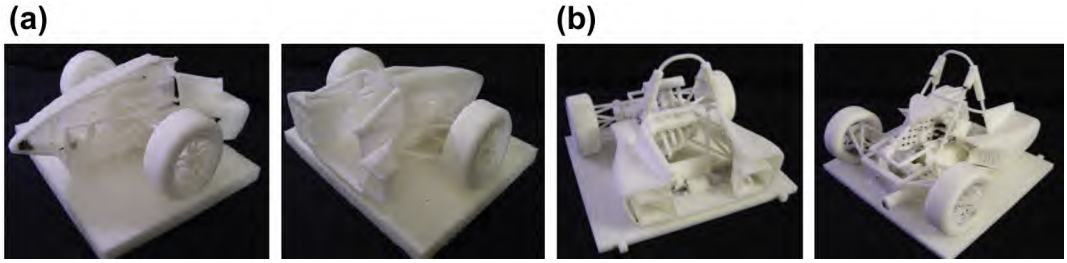


FIGURE 14.34

Physical models of the printed racecar: (a) front half of car and (b) rear half of car (Chang and Long, 2011).

back half of the car, the ease of removing the support material was well worth it. The STL model is then sliced and sent to the Dimension machine for fabrication. It took about 60 hours for the Dimension machine to fabricate the front half of the vehicle, while the more complicated rear half with the engine took about 100 hours to fabricate. The models were then soaked in the liquid solvent to remove the support materials, as shown in Figure 14.34. Figure 14.35 shows the overall process of converting CAD to physical models.

A few closer looks at the physical models are shown in Figure 14.36; comparing with their respective CAD portion demonstrates that the physical models represent the CAD model with an excellent fidelity. The wheel center is shown in Figures 14.36(a) and (b) for the modified CAD model and the final product as printed, respectively. The spokes have been thickened to improve the structural integrity of the printed model. In Figures 14.36(c) and (d), the modified shock can be seen along with the quality of the printed model. The spring on the shock can be clearly seen on the prototype. The front and rear halves of the car are put together on a wood plate for the entire assembly, as shown in Figure 14.37.

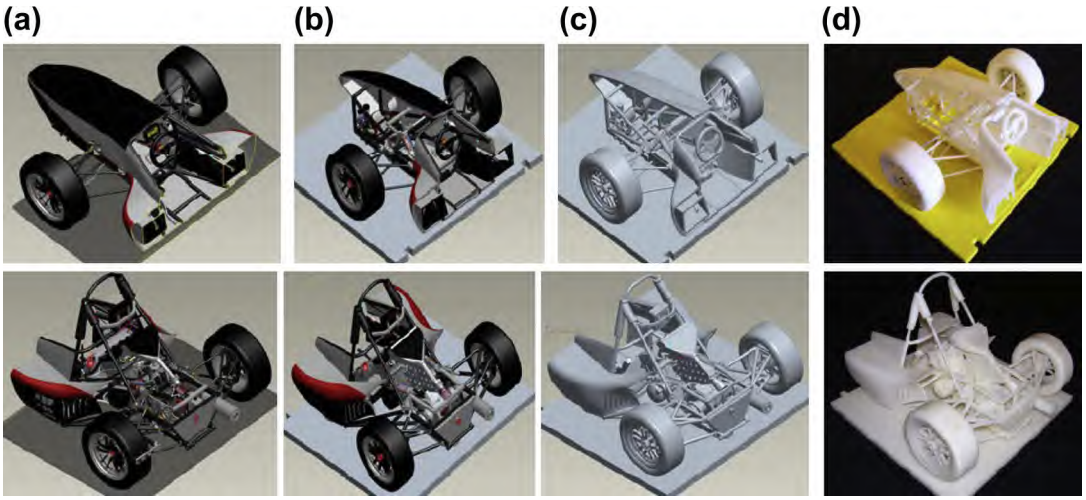


FIGURE 14.35

The overall process of converting CAD to physical models: (a) first column, original geometry; (b) second column, modified CAD model; (c) third column, STL model; and (d) last column, printed model (Chang and Long, 2011).

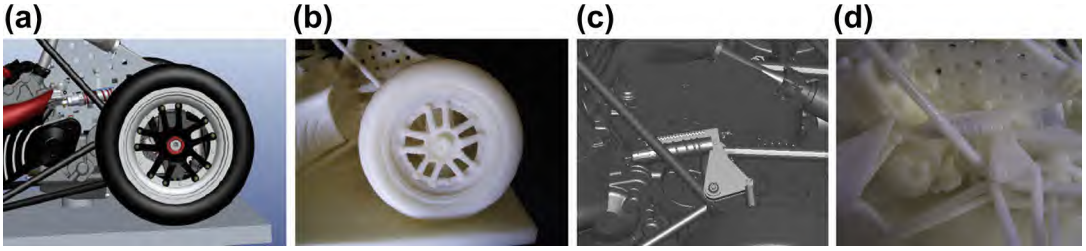


FIGURE 14.36

Closer looks at the CAD and physical models: (a) CAD of wheel center, (b) printed wheel center, (c) CAD of rear shock, and (d) printed rear shock (Chang and Long, 2011).

FIGURE 14.37

The entire racecar on a wood plate.

14.7 SUMMARY

In this chapter, we briefly discussed one of the most exciting and advanced technologies for product design—rapid prototyping (RP) or 3D printing. This technology has the potential to not only reduce the turnaround time in product development, but also to bring significant impact to various areas such as functional prototyping and small production, personal 3D printing, and advancement in medical applications.

We presented solid-, liquid-, or powder-based RP systems; introduced a few emerging technologies capable of fabricating metal objects as strong as machined or cast parts, as well as discussed RPs extended to support microscopic manufacturing.

We also have included prominent RP applications such as industrial design, art, medical science, and bioengineering. One intriguing recent development in RP is tissue engineering; in the foreseeable future, this has a great potential for organ printing and growing cells.

Two case studies were introduced in this chapter—prototype fabrication for a single-piston engine and a Formula SAE racecar—using Dimension 1200sst from Stratasys. We hope these two studies, especially, the racecar one that represents a large-scale complex assembly, provide readers more insight for using rapid prototyping.

After reading this chapter, we hope you will have become more familiar with this new and exciting technology in both the underlying techniques and applications. We hope you are convinced that the RP holds the potential for reducing time and cost in product development and will impact personal 3D printing and medical applications in the near future. After review this chapter, are you considering in the near future buying a personal RP system for your own use?

QUESTIONS AND EXERCISES

14.1. Identify one RP technology or machine of interest to you that has not been included in Section 14.3. Prepare a one-page write up with the following:
- **a.** Source of the information (article, paper, magazine, website, YouTube, etc.)
- **b.** Why is the particular technology of interest to you? Comment on the advantages, disadvantages, pros and cons, and so on, of the RP technology chosen.
- **c.** What are the specific applications of the RP technology and machine you found? List at least one application with sufficient details. Describe how the rapid technology is employed to support such application(s).

14.2. Conduct a case study in the application of RP technology in product development in an industry that was not included in Section 14.5. In your one-page report, please include the following:
- **a.** Name of the company or organization
- **b.** Source of the information (article, paper, magazine, website, YouTube, etc.)
- **c.** What is the nature of the product? What are the challenges engineers face during product development?
- **d.** What are the advantages of using RP for such applications? Discuss cost saving if possible.

14.3. Conduct a case study in the application of RP technology for life science or a medical field that was not included in Section 14.5. In your one-page report, please include the following:

- **a.** Name of the company or organization
- **b.** Source of the information (article, paper, magazine, website, YouTube, etc.)
- **c.** What is the nature of the life science or medical application? What is the RP employed to support such an application?
- **d.** What are the potential impacts that such RP brings to the medical field?

REFERENCES

3D Printing News, February 3, 2012. www.3dprintingnews.co.uk.

3D Printing News, July 8, 2011. www.3dprintingnews.co.uk.

Anatomics, Custom Implants: Titanium Mesh Custom Implant Cranioplasty. www.anatomics.com/content/default.aspx?cat=2&sub=18&tri=15.

Aronson, R.B., 2000. May. It's not just RP anymore. Manufacturing Engineering 124 (5), 98–113.

Asiabanpour, B., Melbye, J., Melbye, V., Jensen, E., Shaw, J., August 4–6, 2008. Freeform fabrication assists forensic scientists in the identification of unknown victims. In: Solid Freeform Fabrication Symposium. Austin, TX.

Beerens, M.M.A., Laeven, P., Poukens, J., Successful jaw implant surgery through advanced virtual planning, Biomedical R&D. Materialize. http://old.materialise.com/Cases/Successful-jaw-implant-surgery?cat=2415332.

Camber, R., January 19, 2005. Tailor-made skin from 'ink' printer. http://menmedia.co.uk/manchestereveningnews/news/s/143230_tailormade_skin_from_ink_printer.

Chang, K.H., Long, T., 2011. Rapid prototyping for complex assemblies. Computer-Aided Design and Applications 8 (3), 357–371.

Chang, K.H., Magdum, S., Khera, S., Goel, V.K., May 2003. An advanced computer modeling and prototyping method for human tooth mechanics study. Annals of Biomedical Engineering 31 (5), 621–631.

Chu, W.S., Kim, S.G., Jung, W.K., Kim, H.J., Ahn, S.H., 2007. Fabrication of micro parts using nano composite deposition system. Rapid Prototyping Journal 13 (5), 276–283.

Chua, C.K., Leong, K.F., Lim, C.S., 2010. Rapid Prototyping: Principles and Applications, second ed. World Scientific Publishing.

Crockett, R.S., Zick, R., August 8–10, 2000. Forensic applications of solid freeform fabrication. In: Proceedings of the 11th Annual Solid Freeform Fabrication Symposium (2000). Austin, TX, pp. 549–555.

Exner, H., Regenfuss, P., Lars Hartwig, L., Sascha Kloetzer, S., Ebert, R., November 20, 2003. Selective laser micro sintering with a novel process. In: Proceeding SPIE 5063, Fourth International Symposium on Laser Precision Microfabrication, 145.

Freitas, V., Queijo, L., Lima, R., January 20–23, 2010. Rapid prototyping of 3D anatomical models to hemodynamic studies. In: Proceedings of the Third International Conference on Biomedical Electronics and Devices, Valencia, Spain.

Hanemann, T., Bauer, W., Knitter, R., Woias, P., 2006. Rapid prototyping and rapid tooling techniques for the manufacturing of silicon, polymer, metal and ceramic Microdevices. In: MEMS/NEMS Handbook Techniques and Applications. Springer, pp. 801–869.

Hart, G.W., 2005. Creating a mathematical museum on your desk. Mathematical Intelligencer 27 (4).

Hollister, S.J., 4, (July) 2005. Porous scaffold design for tissue engineering. Nature Materials 518–524. www.nature.com/naturematerials.

Ikuta, K., Maruo, S., November 25–28, 1998. New Microstereolithography (Super-IEI Process) to create 3D freely movable micromechanism without sacrificial layer technique. In: Proceedings 9th IEEE International Symposium on Micromechatronics and Human Science. Nagoya Congress Center, Nagoya-shi.

Ikuta, K., Maruo, S., 2002. Submicron stereolithography for the production of freely movable mechanisms by using single-photon polymerization. Sensors and Actuators 100, 70–76.

Ikuta, K., Ogata, T., Tsuboi, M., Kojima, S., 1996. Development of mass production micro stereo lithography (Mass-IH process). In: Proceedings of the IEEE International Workshop on Micro Electro Mechanical System (MEMS'96), pp. 301–306.

Jacobs, P.F., 1995. SteroLithography and Other RP&M Technologies. ASME Press.

Kadekar, V., Fang, W., Liou, F., 2004. Deposition technologies for micromanufacturing: A review. Journal of Manufacturing Science Engineering 126 (4), 787–795.

Komaroff, A.F., December 7, 2010. The race to grow new organs. Newsweek Magazine. www.thedailybeast.com/newsweek/2010/12/07/future-of-medicine-growing-new-organs.html.

Liu, Q., Leu, M.C., Schmitt, S.M., 2005. Rapid prototyping in dentistry: Technology and application. International Journal of Advanced Manufacturing Technology 29, 317–335.

Maruo, S., Kawata, S., 1998. Two-photon-absorbed near-infrared photopolymerization for three-dimensional microfabrication. Journal Microelectromechical System 7, 411–415.

Materials World, 2002. Rapid Prototyping—Production of Museum Models by Stereolithography. Materials World 10 (10), 17.

Matthias, S., Hermann, S., Inga, D., 2006. Biomaterials as scaffold for bone tissue engineering. European Journal of Trauma 2, 114–125. Urban & Vogel.

Momi, E.D., Pavan, E., Motyl, B., Bandera, C., Frigo, C., May 2005. Hip joint anatomy virtual and stereolithographic reconstruction for preoperative planning of total hip replacement. CARS: Computer Assisted Radiology and Surgery 1281, 708–712.

Murphy, B., 2002. Sandia's rapid prototyping expertise could help surgeons alleviate chronic back pain. Sandia Lab News, (7). www.sandia.gov/mst/pdf/labnews05-17-02_P7.pdf.

Ngan, E.M., Rebeyka, I.M., Ross, D.B., Hirji, M., Wolfaardt, J.F., Seelaus, R., Grosvenor, A., 2006. The rapid prototyping of anatomic models in pulmonary atresia. The Journal of Thoracic and Cardiovascular Surgery 132, 264–269. © 2006, The American Association for Thoracic Surgery.

Parthasarathy, J., Starly, B., Raman, S., 2011. A design for the additive manufacture of functionally graded porous structures with tailored mechanical properties for biomedical applications. Journal of Manufacturing Processes 13 (2), 160–170.

Rase, W.-D., 2012. Creating physical 3D maps using rapid prototyping techniques. In: Buchroithner, Manfred (Ed.), True-3D in Cartography. Autostereoscopic and Solid Visualisation of Geodata. Lecture Notes in Geoinformation and Cartography. Springer, www.wdrase.de/CreatingPhysical3DMapsRP.pdf.

Roy, S., 2007. Fabrication of micro- and nano-structured materials using mask-less processes. Journal of Physics D: Applied Physics 40, R413–R426.

Séquin, C.H., 2005. Rapid prototyping: A 3D visualization tool takes on sculpture and mathematical forms. Communications of the ACM 48 (6), 66–73.

Siliconrepublic, November 11, 2012. Big ideas give med tech an edge. www.siliconrepublic.com/innovation/item/30399-medtech.

Sun, Q., Chang, K.H., Dormer, K., Dyer, R., Gan, R.Z., November 2002. An advanced computer-aided geometric modeling and fabrication method for human middle ear. Medical Engineering and Physics 24 (9), 595–606.

Swainson, W.K., 1977. Method, medium, and apparatus for producing three dimensional figure product. U.S. Patent 4,041,476.

Visconti, R.P., Kasyanov, V., Gentile, C., Zhang, J., Markwald, R.R., Mironov, V., March 2010. Towards organ printing: engineering an intra-organ branched vascular tree. Expert Opinion on Biological Therapy 10 (3), 409–420.

Waterman, P.J., May 1, 2010. Rapid Tech Firms Aid Medical and Dental Fields. Desk Engineering. www.deskeng.com/articles/aaawxe.htm.

Wolf-Dieter Rase, Creating Physical 3D Maps Using Rapid Prototyping Techniques, Federal Office for Building and Regional Planning. Bonn, Germany www.wdrase.de.

Yang, H., Tsiklos, G., Ronaldo, R., Ratchev, S., 2008. Application of microsterolithography in micro-manufacturing. In: Ratchev, S., Koelemeijer, S. (Eds.), IFIP International Federation for Information Processing, vol. 260, Micro-Assembly Technologies and Applications. Springer, pp. 171–176.

CHAPTER

PRODUCT COST ESTIMATING 15

CHAPTER OUTLINE

15.1 Introduction 789
15.2 Fundamentals of Cost Analysis 792
15.2.1 Elements in the Cost Estimate 793
15.2.2 Type of Costs 794
15.2.2.1 Fixed versus Variable Costs 795
15.2.2.2 Direct and Indirect Costs 797
15.2.3 Overhead Costs 797
15.2.4 Cost-Estimating Techniques 798
15.2.4.1 Qualitative Cost-Estimating Techniques 799
15.2.4.2 Quantitative Cost-Estimating Techniques 801
15.3 Manufacturing Cost Models 802
15.3.1 Manufacturing Cost Elements for In-House Parts 803
15.3.1.1 Processing Cost C_p 803
15.3.1.2 Unit Time t_{unit} 804
15.3.2 Machining Cost Model 804
15.3.2.1 Setup Time T_{su} 805
15.3.2.2 Operation Time t_o 806
15.3.2.3 Non-operation Time t_{no} 809
15.3.2.4 Tooling Cost C_t 809
15.3.2.5 Material Cost C_{mat} 810
15.3.3 Injection Molding Cost Model 810
15.3.3.1 Machine Rate R_m 811
15.3.3.2 Molding Cycle Time t_o 813
15.3.3.3 Mold Cost Estimation 814
15.3.3.4 Material Cost C_{mat} 816
15.3.4 Sheet Metal Stamping Cost Model 816
15.3.4.1 Material Cost C_{mat} 816
15.3.4.2 Processing Cost Rate R_p 818
15.3.4.3 Tooling Cost 819
15.3.5 Assembly Cost Model 819
15.4 Commercial Software for the Cost Estimate 820
15.4.1 CAD-Based Costing Software 820

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00015-6
Copyright © 2015 Elsevier Inc. All rights reserved.

15.4.2 General-Purpose Costing Software 821
15.4.2.1 SEER for Manufacturing 821
15.4.2.2 MicroEstimating 821
15.4.2.3 DFM Concurrent Costing® 821
15.4.2.4 Costimator of MTI System 822
15.4.2.5 aPriori Product Cost Management 822
15.4.2.6 MISys Manufacturing 822
15.4.3 Special-Purpose Costing Software 823
15.4.4 Web-Based Costing Tools 823
15.5 Case Studies 824
15.5.1 Machining Costing Using SolidWorks 824
15.5.2 Sheet Metal Costing Using SolidWorks 829
15.5.3 Cost Estimate for a BWMD Using SEER-DFM 832
15.6 Summary 837
Appendix 15A: Calculations of Material Removed for Standard Features 838
Questions and Exercises 842
References 843

The actual determination of product price is at the heart of business practices and many factors affect product price. One of the key issues is the reaction of the competition and customers. When competition is fierce, the sales force may push for price reductions, claiming they can more than make up lost revenue by increasing volume. Setting a product's price requires knowledge of all the costs associated with the development and manufacture of the product among many other expenses (e.g., sales, support, and administration). The ability to acquire such knowledge and estimate product costs accurately is essential if a business is to survive in the competitive market.

Accurate product cost estimates are important in establishing a good cost control practice and to help with setting competitive price plans. The role of a product cost estimate is, in general, to predict the overall costs likely to be incurred throughout the product development and production phases. Some try to extend the estimate to the product's entire life cycle, including product service, disposal, and so on. The accuracy of predicted costs will determine the effectiveness of the production technique used and, combined with information access, can help the production enterprise achieve corporate goals. The product cost estimating, therefore, is deemed as a key component for a strategic production planning and control system.

The published literature on product cost estimating covers a wide variety of issues ranging from:

- Manufacturing cost estimation of standard mechanical components to cost analysis of highly customized parts and assemblies
- Product and process cost-optimization techniques to specific methods for overhead costing
- Unique approaches for estimation at the conceptual design stage to general costing rules designed for use at a later stage in the design cycle
- Classical costing methods to highly novel cost-estimating techniques.

Several textbooks can be found that offer thorough discussions on some of the subjects—for example, Ostwald and McLaren (2004), Ostwald (1991), and Clark and Lorenzoni (1996)—in addition

to excellent works such as Niazi et al. (2006). Among all those available, the textbook written by Boothroyd et al. (2011), to the author's knowledge, offers the most thorough and in-depth discussion in both manufacturing processes and cost estimation.

It is likely too ambitious to cover all topics just mentioned and provide complete discussion in one single chapter. Therefore, instead of offering superficial or meta-style coverage for a board range of topics in the product cost estimate, this chapter is written to provide design engineers with a fundamental understanding of the subject in the context of product design. In this context, the *product cost estimate* refers to predicting all the costs associated with manufacturing from raw materials to a finished product. To facilitate our discussion and stay focused, we assume that readers are familiar with basic manufacturing processes (e.g., machining, injection molding, sheet metal work); but have little or no knowledge about the product cost estimate.

Therefore, this chapter focuses on introducing cost analysis basics, cost-estimating techniques, and manufacturing cost models that support design engineers who want to obtain quick and accurate estimates for product manufacturing cost during the design phase. With this knowledge, we hope you will be able to produce estimates using hand-calculations, spreadsheets, or software tools in support of product design. A brief review of commercial software tools that support the product cost estimate is provided to make sure readers are aware of them and their applicability to engineering design. We also include three case studies in this chapter to illustrate detailed aspects of manufacturing cost estimates for practical engineering applications using cost-estimating software.

Overall, the objectives of this chapter are to provide basic knowledge of the product cost estimate and to help readers answer fundamental questions such as: How much does it cost to manufacture this part? Is there a better alternative to manufacture the part? Would it be less or more expensive to do so?; to familiarize readers with practical applications of the cost-estimating techniques and to be able to apply them adequately for support of product design; to introduce readers to existing commercial product cost-estimating software.

We hope this chapter helps you develop a cost-sensitive mind-set in carrying out product design.

15.1 INTRODUCTION

Business enterprises are facing a globally competitive market to an extent not experienced before. Factors, such as globalization and mass customization, put extra pressure on a business to survive and remain profitable at the same time. Whereas an innovative approach and a new product development process may attempt to deal with some issues (e.g., flexibility and product quality), they may still be time-consuming and less cost-effective. This competitive pressure forces manufacturers to make more products in a shorter time and better quality, yet at a lower cost.

To succeed in this environment, businesses need to have an accurate estimate of product design, development, and production costs. In a competitive situation, if a company's estimate of its costs is unrealistically low (underestimate), then it may obtain an order but risk profit loss and a blow to operational targets. On the other hand, overestimating costs may cause the company a more profound strategic damage—the loss of customer goodwill and market share. All the preceding highlights the ever-increasing importance of devising methods to accurately forecast the cost for a new product in the early design and development phases.

The accuracy of cost estimates is essential to the survival of an organization. The relationships between the over- and underestimates of product cost can be represented by the Freiman curve shown

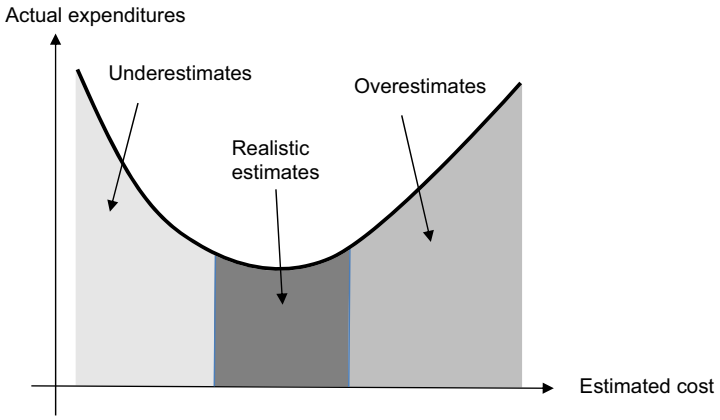


FIGURE 15.1

The Freiman curve (Asiedu and Gu, 1998).

in Figure 15.1 (Asiedu and Gu, 1998). The graph shows that the greater the underestimate, the greater the actual expenditure, the greater the overestimate the greater the actual expenditure, and the most realistic estimate results in the most economical project cost.

When costs are underestimated, initial plans for staffing, scheduling, machine processing, tooling, and so on are not achievable. In response, there is reorganization, replanning, and possibly the addition of personnel and equipment (Bouaziz et al., 2006). Such things tend to incur costs that were not originally budgeted for, eventually resulting in an increase in costs. On the other hand, when costs are overestimated, rather than resulting in greater profits, the overestimate reflects a Parkinson's law application (Asiedu and Gu, 1998): the money is available so it must be spent. Unless firm management control is exercised, there is a self-fulfilling prophesy and it will be virtually impossible to reduce costs.

A product cost estimate that predicts all the costs associated with manufacturing a product from raw material to a finished product is a challenging task. Product cost-estimating techniques aim to predict costs early and accurately before actual production takes place and sometimes even before the design cycle. This is mainly because most product costs sustained later in the production life cycle are determined during the conceptual design phase (Cavalieri et al., 2004). Many researchers (e.g., Roskam, 1985) have emphasized the importance of cost estimation at the early design stages when 70 to 80 percent of total product cost is determined (Figure 15.2). In addition, the possibility of influencing cost during the design phase is much higher than during the other phases, while at the same time the modification cost is substantially smaller compared with other stages in the life cycle. Product modifications and process alterations are exponentially more expensive the later they occur in the product development cycle (Bakerjian, 1992). Thus, good cost estimation as early as possible assists with controlling the parameter of cost, which subsequently implies that the enterprise's performance and effectiveness will be significantly and positively influenced (Chryssolouris, 2006).

Unfortunately, product cost estimating is not an exact science and in the best of circumstances will provide only an approximation of the costs that actually will be incurred. This is especially true during

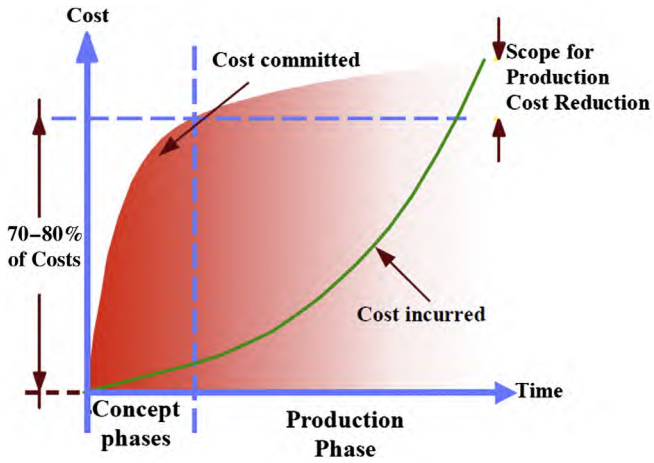


FIGURE 15.2

Cost commitment curve (Roskam, 1985).

the product design phase when available information is limited since the product is not fully defined, imposing a great obstacle for cost evaluation.

Developing meaningful estimates for design alternatives at the design phase is crucial and yet a nontrivial task (Meisl, 1988). There is a high degree of uncertainty attached to the final estimate, as depicted in Figure 15.3 (Roy et al., 2002; Rush and Roy, 2001). The figure illustrates that during the early stages of development, the high degree of uncertainty mainly is a result of the lack of product detail and consequently a large estimating error. As product detail takes shape, the unknowns diminish and the expected error range is reduced.

Providing accurate product cost estimates in the early stage requires adequate cost data and knowledge about cost-estimating techniques. Cost data include cost of similar products previously manufactured, cost of the manufacturing process and equipment involved, among many other things. Such data are usually well documented in business and must be provided to design engineers for cost estimation.

There is no doubt that the product cost estimate is a broad topic, which is very difficult if not entirely impossible to master in a short time, let alone to introduce it in one single chapter. Furthermore, design engineers on a product development team usually are given a task of estimating product manufacturing cost to determine a set of cost-effective manufacturing processes for production. With that, a design engineer must understand cost data and cost-estimating techniques in order to estimate product manufacturing cost during the early product design stage with acceptable accuracy. He or she must have a cost-sensitive mind-set when conducting product design and be able to answer common questions such as: How much does it cost to manufacture this part? Is there a better alternative for manufacturing the product?

Therefore, in the context of product design, we narrow the definition of the *product cost estimate* to predicting all the costs associated with manufacturing a product from raw material to a finished product. We will devote most of the effort in this chapter to introducing costs associated with

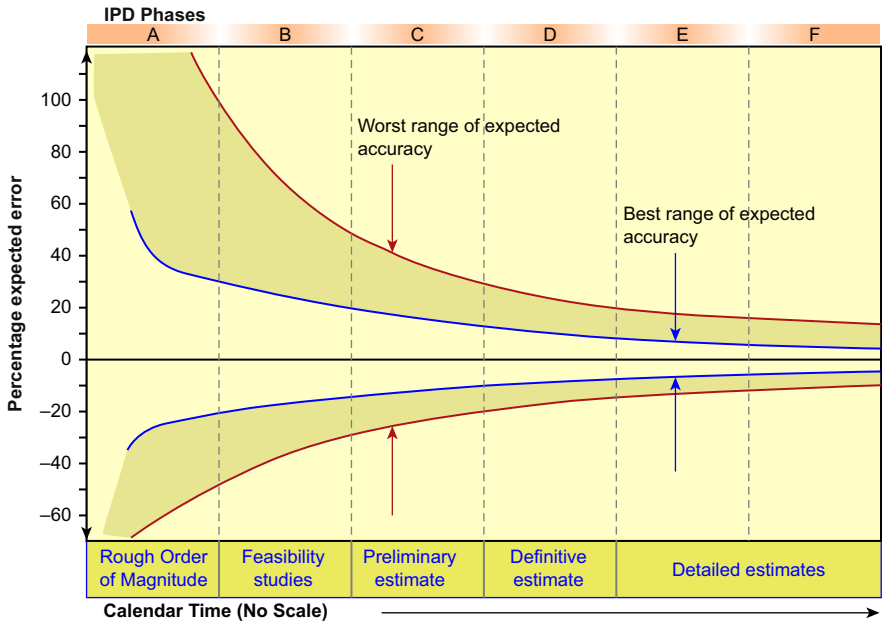


FIGURE 15.3

Expected error range of estimates (Roy et al., 2002).

manufacturing, manufacturing cost models, and widely accepted cost-estimating methods; in addition, fundamentals in cost analysis are covered to provide readers basic knowledge in learning the subject.

This chapter will start by briefly discussing basics in cost analysis in Section 15.2, which aims at providing fundamental knowledge about the product cost estimate in a broader sense, including cost-estimating techniques. With this basic knowledge, we will narrow our scope to estimating product manufacturing costs. Starting with Section 15.3, we will introduce cost models for the most common manufacturing processes, followed by Section 15.4 in which we will introduce commercial cost-estimating software. We also will include in Section 15.5 a few case studies as examples to illustrate practical aspects of product cost estimates for engineering applications. The studies include machining costing and sheet metal costing using SolidWorks, and cost analysis for a bicycle wind measurement device (BWMD) using SEER-DFM software, a leading industrial cost-estimating software.

The theme of this chapter is product manufacturing cost estimates during the design phase.

15.2 FUNDAMENTALS OF COST ANALYSIS

In this section, we will provide a brief overview on cost analysis in a more general sense in the hope of offering readers a general understanding on the various aspects of the subject. To stay in focus, we assume a company that designs, develops, manufactures, and sells products, instead of addressing business practices in general. We will discuss common cost elements in the product cost estimate, type of costs, and cost-estimating techniques to offer readers a basic understanding of cost analysis.

15.2.1 ELEMENTS IN THE COST ESTIMATE

Conventionally, product price is built from the bottom up, as shown in Figure 15.4. The major cost elements involved in product manufacturing cost consist of direct labor, direct material, and tooling, which determine the primary cost of production. Also involved in determining manufacturing cost is the factory expense, including utilities, maintenance, supplies, factory indirect labor, and so on—often referred to as manufacturing indirect cost. In addition to manufacturing cost, general and administrative cost, engineering cost, and sales force expenses are added to manufacturing costs for a total product cost. The general and administrative cost comprises depreciation, taxes, office staff, purchasing, and so forth. Engineering cost includes R&D, product design, development, prototype, and test. Finally, the selling price is established by adding a profit to the total cost. These costs—classified as general, engineering, and sales—are usually handled by overhead practices. Many refinements can be undertaken, and most of them usually are worth the effort since costs can be more accurately distributed to the product for eventual recovery. Another cost that could be substantial for a manufacturing company is the capital investments (e.g., equipment, building, and land); these are often recovered from the profits of several product lines. Some of the investments, especially manufacturing equipment, are recovered by building in the processing rate of using the equipment for manufacturing parts.

Note that pricing from the bottom up, as shown in Figure 15.4, is an ordinary practice. Under competitive conditions, product price may be limited by the market. In that situation, the price and costs are set from the top down. Design and manufacture to profit and cost principles are a popular engineering practice.

Another important cost category (Dieter, 1991) is working capital—the funds that must be available in addition to fixed capital (e.g., equipment, facility, and building) and land investment—to get a project or product development started and to provide for subsequent obligations as they come due. The working capital consists of raw material on hand, purchased parts and hardware from vendors, semifinished product in the process of being manufactured, finished product in inventory,

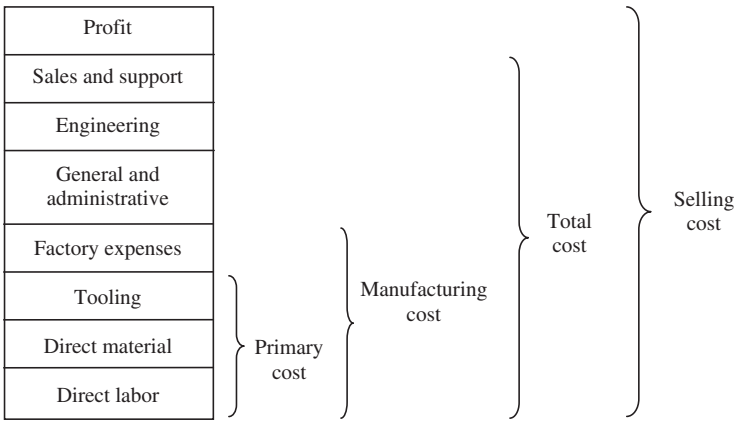


FIGURE 15.4

Elements of cost that establish the product's selling price.

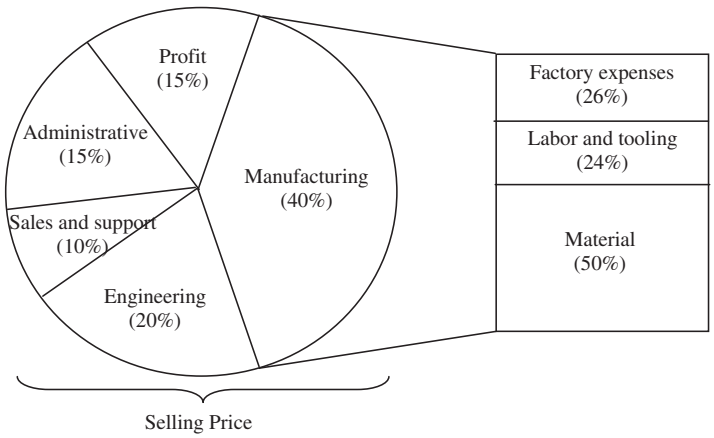


FIGURE 15.5

A typical cost breakdown for a manufacturing scenario (Black, 1991).

accounts receivable, and cash needed for day-to-day operations. The working capital is tied up during the life of the factory, but it is considered to be fully recoverable at the end of the life of the project.

Every manufacturing company that develops and sells products has some way to assign the various costs associated with manufacturing products. Figure 15.5 shows a representative example of how the costs of a manufactured product are broken down. The numbers are just typical values collected by Black (1991) and should by no means be considered definitive. However, it is of interest to note that the total cost associated with a product is often dominated by the manufacturing cost, which in turn is dominated by material cost in many cases. For some manufacturing processes (e.g., injection molding) tooling cost is often dominant.

In general, a cost estimate for a product undergoing redesign requires a different treatment than a new product. The components of the redesigned product are compared to the existing product and classified as changed, added, or identical. The costs of the unchanged parts are found from the records. Estimates are made for the new or modified parts. In either case, manufacturing cost must be estimated. In industry, redesign is much more common than developing a brand new product.

15.2.2 TYPE OF COSTS

Costs can be broken down into many different categories. There are several common ways to categorize product costs such as nonrecurring and recurring, fixed and variable, direct and indirect (Dieter, 1991). Nonrecurring costs, which are usually called capital costs or capital investment, include depreciable facilities (e.g., manufacturing equipment and tools) and nondepreciated capital investment such as land. Recurring costs are a direct function of manufacturing or product development costs that occur over and over again. Fixed costs are independent of the rate of production of goods. Variable costs change with production rate. Direct costs are directly assigned to a particular product line or part. Indirect costs cannot be directly assigned to a product but must be spread over an entire factory. In this subsection, we further discuss the fixed versus variable costs and direct versus indirect costs.

15.2.2.1 Fixed versus Variable Costs

Fixed costs are those that remain relatively constant over the product's life cycle, including design, development, prototype, test, R&D, sales and support, and disposal. Fixed costs are independent of production volume or output and include elements such as depreciation, taxes, insurance, interest on invested capital, general supplies, rental of equipment, sales, technical services, and administrative expenses. Fixed costs also arise from prediction about where the company wants to be in the future. Equipment purchased now may allow reduction of labor costs in the future or may provide for product improvement or diversification. A concept that provides a rough estimate of the investment cost for a new product is the turnover ratio: turnover ratio = annual sales/total investment.

To depict business operating performance, turnover ratio always should be compared to the industry average because it varies greatly between different industries. In capital-intensive industries (e.g., steel, auto, and heavy manufacturing companies), the turnover ratio is typically less than one, while in retail and services companies it may be more than ten. In addition, the ratio reflects a range of turnover values consistent within the industry. For example, a low turnover ratio relative to the industry average might imply that the business does not generate a sufficient volume of sales given its investment.

In addition to equipment investment, maintenance costs must be considered. Equipment maintenance costs are the costs incurred in operating and maintaining equipment in the context of product manufacturing. They include maintenance labor, fuel, power costs, supplies, spare parts, repairs, insurance, taxes, and overhead and usually increase with time. A common way to determine when an item should be replaced is to use a plot of average maintenance and capital costs versus the economic life of the equipment. Figure 15.6 shows a typical plot of this type.

Variable costs, on the other hand, are a function of the level of production or activity. As production levels increase, certain (variable) costs increase. Variable costs are usually proportional to the number of units made. In other words, they are maximized when a plant is operating at full capacity. Variable costs, which usually include labor, material, tooling, utilities, and operating costs, are much more relevant to designers during product cost estimation.

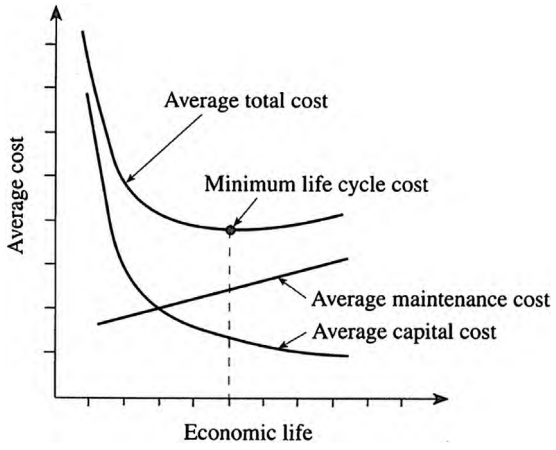


FIGURE 15.6

Asset life-cycle cost (Fabrycky and Blanchard, 1991).

Depending on the type of manufacturing, particularly in labor-intensive processes, labor costs can be the dominant cost factor. Generally, it is advisable to make a table of the labor requirements that includes skills required for each operation, labor rate for that skill level, supervision required, and support personnel. It should be noted that benefits (e.g., health, dental, pension) must be paid also. These typically amount to 20 to 40% of the wage rate.

Material cost, as a common practice, is the cost of the raw materials less the value of the scrap; that is,

$$C_{\text{mat}} = W_o R_{\text{mat}} - (W_o - W_f) R_s \tag{15.1}$$

where W_o is the initial stock weight, W_f is the final part weight, R_{mat} is the cost rate of stock material (e.g., \$/lb.), and R_s is the rate of scrap (\$/lb.). Note that Eq. 15.1 represents a common practice. For some other manufacturing processes (e.g., machining), material cost is calculated by

$$C_{\text{mat}} = V_w \, \rho \, R_{\text{mat}} \tag{15.2}$$

where V_w is the volume of the workpiece and ρ is the weight density of the workpiece. Often, prices for materials fluctuate on a daily basis. Material cost rates are routinely published—see, for example, Get-A-Quote.net (www.get-a-quote.net).

In addition to material costs, standard parts (e.g., nuts, bolts, springs, bushings) can be found in suppliers' catalogs. For example, the Thomas Register is an online database of products made by more than 150,000 companies (www.thomasregister.com). Tooling costs are costs for cutting tools, dies, molds, jigs, and fixtures used specifically for the manufacture of the product. For some manufacturing processes (e.g., injection molding) tooling costs are usually dominant in part cost.

If both fixed and variable costs are known, the total cost can be computed by the following relation:

$$C_T = C_{\text{fix}} + Q \, C_{\text{var}} \tag{15.3}$$

where the total cost, C_T (\$), is the sum of the fixed cost, C_{fix} (\$), and the product of the production quantity, Q, and the variable cost, C_{var} (\$/part). The fact that variable costs depend on the volume of production, and fixed costs do not, leads to the idea of a break-even point (Figure 15.7).

Determining the production lot size that will exceed the break-even point and produce a profit is an important consideration. There are many factors to be considered, but a common decision associated with economic lot size is how to allocate production among different machines, plants, or processes of

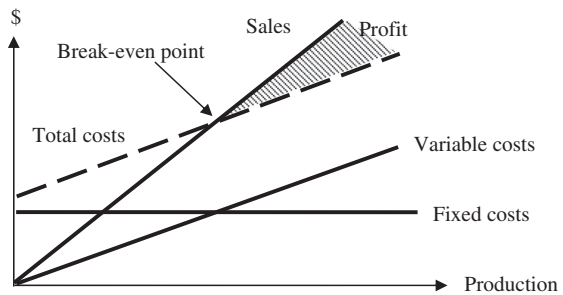


FIGURE 15.7

Break-even curve showing relationship between fixed and variable costs and profit.

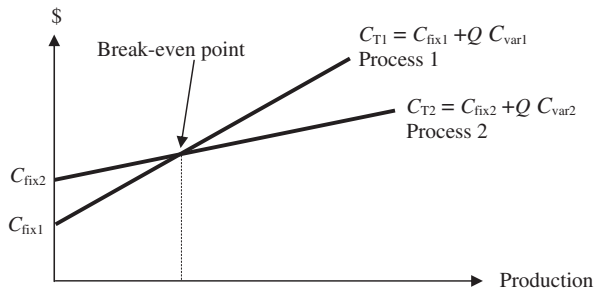


FIGURE 15.8

Comparing two processes based on fixed and variable costs.

various efficiencies, or cost structure, to make a product at minimum cost. Furthermore, if an alternative manufacturing process is considered, a graph like that in Figure 15.8 is useful in terms of selecting a more cost-effective process for a given production quantity; in this case, the break-even point refers to the production quantity that requires the same manufacturing cost between two competing processes.

15.2.2.2 Direct and Indirect Costs

In addition to the fixed/variable classification, costs can be categorized as direct and indirect. Direct costs are those that can be traced to a specific product such as direct labor and material costs. Indirect costs are those that cannot be traced to a specific product but are still required for production. Some examples of indirect costs include factory expenses, product design and engineering costs, general and administrative costs, and cost of sales.

Sometimes indirect costs are all lumped into one sum called *overhead,* which is simply a convenient tool for covering the quantities that are fuzzy or too difficult to assign as a direct cost. It can be accounted for in the direct cost formula by applying an overhead rate to direct cost. Depending on the level of detail of the cost estimation model, overhead can be decomposed into various direct costs and predicted more accurately. The accurate calculation of overhead (also called burden) is important for cost estimation and will be discussed in further detail next.

15.2.3 OVERHEAD COSTS

For a manufacturing company, overhead costs are all of the indirect expenses associated with running the manufacturing company (Groover, 2001). Overhead can be subdivided into two categories: manufacturing and corporate. Manufacturing overhead covers the indirect fixed costs associated with running the factory for product manufacturing, whereas corporate overhead deals with the other administrative expenses. Total overhead typically is described in units of dollars per year and represents all indirect expenses the company experienced in one year. This total yearly overhead value can be accounted for in determining the total manufacturing cost of a product by appropriately absorbing it into direct production costs. This can be accomplished using some form of transformation that converts the total overhead into an hourly cost. This is typically done by dividing overhead costs by some appropriate base.

Groover (2001) suggests using the total direct labor cost as the base; this is the most common method used in industry. To obtain overhead rates, historical cost data from previous years are used to calculate total direct labor and indirect overhead costs. Then the labor rates are computed as follows:

$$OH_f = \frac{C_{OH_f}}{C_\ell} \tag{15.4a}$$

$$OH_C = \frac{C_{OH_C}}{C_\ell} \tag{15.4b}$$

where the factory overhead rate, OH_f (%), is simply the annual cost of factory overhead expenses, C_{OH_f} ($/year), divided by the annual direct labor costs, C_ℓ ($/year). The corporate overhead rate, OH_c (%), is calculated in a similar manner, except using the annual corporate overhead expenses, C_{OHc} ($/year), as the numerator.

The preceding overhead rates can be further subdivided into individual rates. For example, the factory overhead rate, OH_f, can be divided into labor overhead, OH_ℓ, and machine overhead rates, OH_m, by using only the total costs of labor and machines, respectively, as the numerator. More specifically, the factory overhead rate attributed to running a particular machine, OH_m (%), would be computed by dividing the total annual cost of running the machine by the annual direct labor costs, C_ℓ ($/year).

Once the appropriate overhead rates have been estimated based on historical cost data, the per-piece cost of overhead required to manufacture the product can be accounted for by adjusting the direct costs by these rates. For instance, Groover calculates the total hourly processing cost rate as:

$$R_p = R_\ell\left(1 + OH_\ell\right) + R_m\left(1 + OH_m\right) \tag{15.5}$$

where the total hourly cost, R_p ($/hr.), is the sum of the direct labor and machine-processing rates, R_ℓ, and R_m, respectively ($/hr.), each adjusted by their own associated indirect overhead rates. The appropriate overhead rates are the overhead rate for labor, OH_ℓ (%), and the overhead rate for machine operation, OH_m (%).

15.2.4 COST-ESTIMATING TECHNIQUES

Estimating a product cost can be a substantial task. Without much effort, a rule of thumb—the 1-3-9 rule suggested by Ullman (2003)—is widely accepted for an initial cost estimate. The 1-3-9 rule says that for a product of $1 in material would require $3 to manufacture, and would sell for $9 to be profitable. The cost of material in general includes raw materials, purchased parts, and scrap. Manufacturing cost includes labor for manufacturing and assembly, related overhead, and packaging materials for shipping. The sales price incorporates salary and benefits for design, finance and accounting, utilities and building costs, and all other overhead needed to run the company, plus a profit margin, as illustrated in Figure 15.4.

This is certainly a rough guideline for companies with products that are manufactured primarily in-house at a high volume. It will vary based on the product, the company, and the production volume. There are two aspects to pricing a product. The first is to be sure that all costs are included, not just the obvious costs of labor, materials, and overhead. The second aspect concerns the business strategy of setting the price based on the volume price relationship and the estimate of the product's market potential.

In the literature, a variety of different cost-estimating techniques can be found, while numerous classification schemes have been proposed for the taxonomy of these cost methods. However, it seems that there is common consensus concerning classification of cost-estimation methods in quantitative and qualitative techniques (Chryssolouris et al., 2008; Niazi et al., 2006).

15.2.4.1 Qualitative Cost-Estimating Techniques

Qualitative cost-estimating techniques are primarily based on a comparative analysis of a new product with the products that have been manufactured previously in order to identify similarities in the new one. In this case, the cost of similar products manufactured before is set as a basis, and the cost of the new product can be estimated by making adequate adjustments from the basis. The adjustments are determined by the level of similarity between new and existing products. This kind of technique is often referred to a case-based reasoning; it is suitable for a cost estimate at an early stage when the product has not been quite realized.

Therefore, one key element in qualitative techniques is to identify similarities between new and existing products. Identified similarities help to incorporate past data into the new product so that the need to obtain the cost estimate from scratch is greatly reduced. In that sense, past design and manufacturing data, or the previous experience of an estimator, can provide useful help to generate reliable cost estimates for a new product that is similar to a previous design. Sometimes, this can be achieved by making use of past design and manufacturing knowledge encapsulated in a system based on rules, decision trees, and so on.

Products are being compared at all levels, from entire products to individual subsystems, components, or even solid features; of course, this depends on the availability of cost data for existing products, development stage of the new product, purpose of the cost estimate, and the desired accuracy of the estimate.

At product or subsystem level, one common method is regression analysis, which often provides an efficient way to predict costs for new products by using historical cost data. One such example is for aircraft development, where typically mass relates to the cost of production. That is, as the weight of the aircraft increases, so does the cost of producing it. This particular relationship is often described as linear, as illustrated in Figure 15.9.

In this hypothetical example, the points on the graph represent the relationship of cost to mass for different aircraft. The line traversing the points represents a linear relationship; that is, as the mass increases so does the cost. Using relatively simple algebra, it is possible to derive a formula to determine a mathematical relationship for cost versus mass. For the graph in the figure, the equation $y = a + bx$ is used to describe the line of best-fit between the points. With such relationships defined, it is possible to predict the cost of a product in a speedy, systematic fashion with a relatively limited amount of information. However, the cost estimated is less certain. Such methods may be suitable for the product cost estimate at an early product development stage.

Another technique, often referred to as the analogical technique, is to identify and quantify similarities between new and existing products and to use the product cost of the existing one as a base to estimate cost for the new product. The overall weighted similarity, S_o, can be quantified by

$$S_o = \frac{\sum_i (w_i f_i)}{\sum_i w_i} \tag{15.6}$$

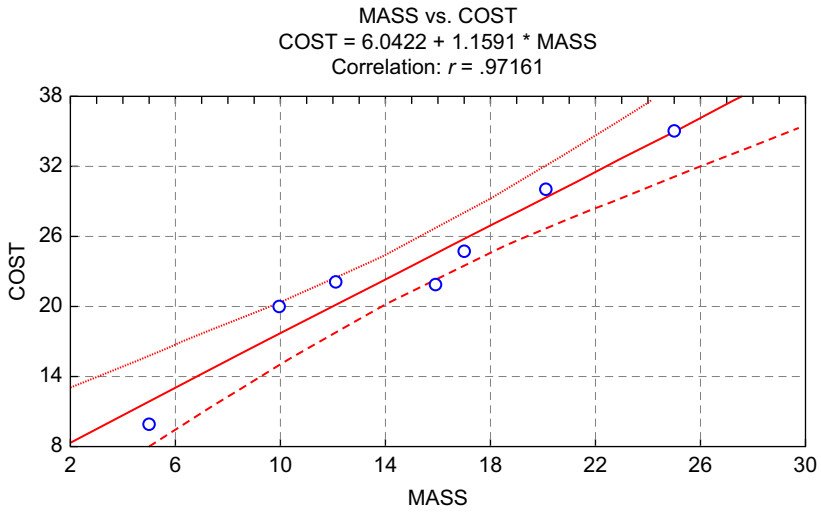


FIGURE 15.9

Simple linear relationship between mass and cost (Rush et al. 2003, www.galorath.com).

where f_i is the value of the ith similarity factor assigned and w_i is the weighting factor of the ith similarity factor.

This technique can be applied to product at more detailed levels. For example, geometry similarities between a new part and existing part(s) can be identified and quantified for cost estimates. If both parts are machined, then the similarities between them are compared in terms of, in general, part material cost, part size (or weight), number of features, feature size, and complexity of features. As an example, consider the two torque tubes shown in Figure 15.10: Tube A is a new design that requires a cost estimate and Tube B is a previously manufactured one for which cost information is available. Assuming both tubes are made of the same material (Aluminum Alloy 2014) and both are machined, they are similar and hence reasonable candidates for using the analogical technique for a cost estimate.

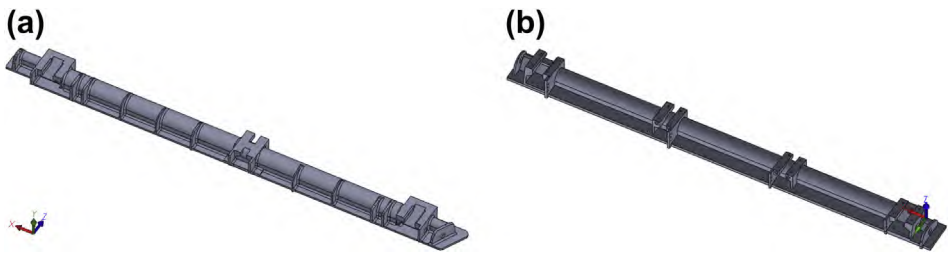


FIGURE 15.10

Torque tubes: (a) Tube A, new design (80.56 in. long and 29.27 lb.); and (b) Tube B, previously manufactured (73.01 in. long and 22.09 lb.).

As shown in Figure 15.10, we learn that the lengths of Tubes A and B are 80.56 and 73.01 in., respectively and that the stock material volume ratio is roughly 1.10 (i.e., 80.56/73.01). Therefore, stock material cost is expected to be 10% more for the new part. Tubes A and B have three and four brackets, respectively; thus, Tube A will take 75% of the machining time it took for Tube B's bracket machining. In addition, Tube A has nine fins on the exterior cylindrical surface of the tube and Tube B has none. Machining time for the exterior cylindrical surfaces for Tube A will be significantly more than that of Tube B, assuming that the factor is 2. These similarity factors are then weighted by a 2:1 ratio between material and machining costs. Thus, the overall similarity that is directly related to part cost between Tubes A and B can be calculated using Eq. 15.6 as

$$S_o = [w_1 f_1 + w_2(f_{2b} + f_{2s})]/(w_1 + w_2) = [2 \times 1.10 + 1 \times (0.75 + 2)]/(2 + 1) = 1.66$$

If the cost of Tube B is \$2000, the cost of Tube A can be estimated as \$3320 using the analogical technique.

15.2.4.2 Quantitative Cost-Estimating Techniques

Quantitative techniques, on the other hand, are based on a detailed analysis of a product design, its features, and the corresponding manufacturing processes instead of simply relying on past data or knowledge of an estimator. Costs are, therefore, either calculated using an analytical function of certain variables representing different product parameters, or as the sum of elementary units representing different resources consumed during a whole production cycle, of a given product. Although these techniques are known to provide more accurate results, their use is normally restricted to the later phases in the design cycle due to the requirement for a detailed product design. Figure 15.11 shows how these estimating techniques can be used relative to the various phases of a typical product development process.

In general, quantitative techniques assume that a detailed bill of materials (BOM) is available, which is usually the case during the detailed design phase. In addition, part materials and manufacturing and assembly processes for individual parts and subsystems are assigned. With such

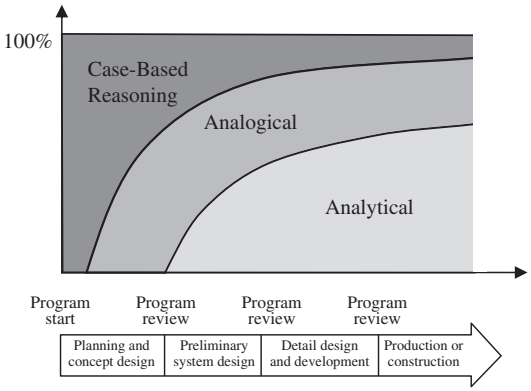


FIGURE 15.11

Use of cost-estimating techniques during product development phase (revised from Fabrycky and Blanchard, 1991, p. 148).

more complete product information, product cost can be estimated by decomposing a product into elementary units, operations, and activities that represent different resources consumed during the production cycle; the cost can be expressed as a summation of all these components. Several approaches have been developed for analytical cost estimates (Niazi et al., 2006).

One of the approaches in this category is the operation-based approach; it allows the estimate of manufacturing cost to be a summation of the costs associated with the time of performing manufacturing operations, nonproductive time, and setup time. For example, the cost model proposed by Jung (2002) estimated the manufacturing cost by considering three different times, including setup time, operation time, and non-operation time. The machining cost, C_m, was given as

$$C_m = R_m(T_{su}/Q + t_o + t_{no}) \tag{15.7}$$

where R_m is the machining rate ($/hr.), T_{su} is the setup time, t_o is operation time, t_{no} is non-operation time, and Q is the batch size.

In general, analytical techniques provide much more accurate cost estimates, but require a lot more effort in acquiring cost data and carrying out cost calculations.

15.3 MANUFACTURING COST MODELS

In this section, we discuss the key topic of the chapter: manufacturing cost models, which may be directly applicable to the cost estimate tasks at hand for engineers or designers. Cost models for widely employed product manufacturing processes, including machining, injection models, sheet metal stamping, and assembly, are presented. To stay focused, we assume that readers are familiar with these manufacturing processes. If this is not the case, please review manufacturing process textbooks (e.g., (Kalpakjian and Schmid, 2010; Boothroyd et al., 2011) to learn more about these common processes before reading this section.

Also notice that every manufacturing company or factory has developed its own manufacturing cost models that are tailored to meet specific needs and are complied with practices at shop floors. In general, manufacturing cost consists of component cost, assembly cost, and overhead cost, as shown in Figure 15.12. Some components (e.g., standard parts) are purchased externally and some are manufactured in-house. Manufacturing cost for an in-house part can be further divided into material cost,

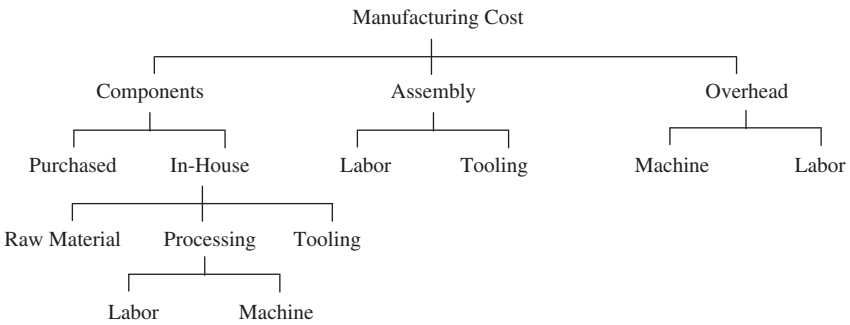


FIGURE 15.12

Manufacturing cost breakdown.

processing cost, and tooling cost. Processing cost includes labor and machinery. Assembly cost consists of labor and tooling. Overhead cost, in general, is divided into machine and labor overhead.

Manufacturing cost models for in-house parts are the focus of this section. Shown here are representative examples that should enable readers to understand the major cost factors in estimating products' manufacturing cost, the data required, and the steps to acquire such an estimate. Some of the details presented are collected from textbooks (e.g., Boothroyd et al., 2011) and research literature (e.g., Dewhurst and Kuppurajan, 1987; Lovejoy et al., 2010). Some data in this section were obtained empirically and some cost figures may need adjustment; for instance, inflation must be considered to bring these figures up to date. This information should by no means be considered definitive.

15.3.1 MANUFACTURING COST ELEMENTS FOR IN-HOUSE PARTS

The total cost of an in-house manufactured part is the sum of the processing cost C_p; the tooling cost C_t, and the material cost C_{mat}—that is,

$$C_u = C_p + C_t + C_{\text{mat}} \tag{15.8}$$

where C_u is the total unit (per piece) cost. The processing cost C_p is an important part of the unit production cost. It depends on the unit time t_{unit}; the cost rates of the machine R_m (machine rate, \$/hr.); labor R_ℓ (labor rate, \$/hr.); and overhead.

15.3.1.1 Processing Cost C_p

The processing cost can be obtained by

$$C_p = R_{\text{p}}\, t_{\text{unit}} \tag{15.9}$$

where R_p is the processing rate (\$/hr.), which is the sum of cost rates of the machine R_m and direct labor R_ℓ with overhead (i.e., like those in Eq. 15.5); that is,

$$R_p = R_\ell(1 + OH_\ell) + R_m(1 + OH_m) \tag{15.5}$$

In general, the machine rate R_{m} depends on the equipment purchased cost C_e, the payback period Y_p (years), and the number of work shifts N_s. One standard way of calculating the machine rate R_m is

$$R_m = \frac{C_e}{Y_p N_s T_a} \tag{15.10}$$

where T_a is the number of hours that the machine is available per shift per year, which can be calculated by

$$T_a = E_{ff}\ 2000 \text{ hours} \tag{15.11}$$

where E_{ff} is plant efficiency—representing the percentage of time the plant (and manufacturing line) is up and running. We assume that the standard work hours per year is 2000 with 8 hours per workday and five workdays per week. Therefore, for a plant with 70% efficiency, the number of hours is $T_a = 0.7 \times 2000 = 1400$ hrs.

Note that the machine overhead rate includes costs (e.g., power, maintenance and supplies, services) that are determined on an annual basis and converted into per hour costs. The labor overhead

rate includes costs, such as salaries of factory supervisor and support staff, plus fringe benefits. Overall, the overhead rates must be sufficient to recover factory expenses, as shown in Figure 15.4.

EXAMPLE 15.1

For a batch of machined parts that require a CNC mill purchased for $250,000, calculate the machine rate. Two shifts are planned for a plant that has an average 70% efficiency rate in a 5-year period. Assume that the machine is to be paid back within 5 years. Also, if the labor rate is $50/hr. and the overhead rates for labor and machinery are 120% and 80%, respectively, calculate the processing rate R_p.

Solution

The machine rate R_m can be determined by using Eq. 15.10 as

$$R_m = \frac{C_e}{Y_p N_s T_a} = \frac{250{,}000}{5(2)(1400)} = \$17.9/\text{hr}$$

From Eq. 15.5, the processing rate is

$$R_p = R_\ell(1 + OH_\ell) + R_m(1 + OH_m) = 50 \times (1 + 1.2) + 17.9 \times (1 + 0.8) = \$142.2/\text{hr}.$$

15.3.1.2 Unit Time t_{unit}

The production time for a unit t_{unit} is the sum of the setup time per unit (T_{su}/Q), operating time t_o, and non-operating time t_{no}; that is,

$$t_{\text{unit}} = T_{su}/Q + t_o + t_{no} \tag{15.12}$$

The setup time T_{su} is the sum of the time required to set up machines or equipment that manufacture the part; for example, the setup time may involve getting a lathe and a drill ready if a part consists of turning features and holes. In Eq. 15.12, Q is the number of parts in a batch job, t_o is the operating time directly contributing to part manufacture, and t_{no} is the non-operating time not contributing to part manufacturing. The non-operating time in general includes the time the operator spends loading and unloading the workpiece on the machine, downtime due to machine malfunction or tool failure, among others.

15.3.2 MACHINING COST MODEL

According to a survey of the metal working industry (Jung, 2002), the turning operation has 24.9% of incidents of metal cutting, milling has 20.2%, and drilling has 28.2% of the work. The sum of these three figures is 73.3% of machining work; these are the primary metal-cutting operations that cover a large portion of the work. Therefore, in this subsection, we intend to include all three major cutting operations.

Machining cost calculation is mainly based on machining processing cost C_p, tooling cost C_t, and material cost C_{mat}, similar to that of Eq. 15.8. Machining time is composed of setup time, operation time, and non-operation time. Operation time and non-operation time are usually proportional to the quantity of production. The setup time is proportional to the number of settings per batch. Therefore, setup time per part is the total setup time divided by a batch size Q; this cost can be estimated by

$$C_u = C_p + C_t + C_{\text{mat}} = [R_m(1 + OH_m) + R_\ell(1 + OH_\ell)](T_{su}Q + t_o + t_{no}) + C_t + C_{mat} \tag{15.13}$$

15.3.2.1 Setup Time T_{su}

Setup time for machining commonly can be calculated as

$$T_{su} = \sum_i t_{a_i} + \sum_i \sum_j t_{b_{ij}} \tag{15.14}$$

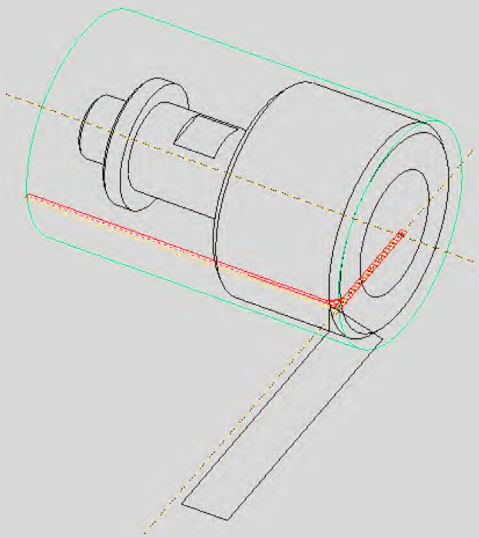
where t_{a_i} is the setup time for the ith machine, $t_{b_{ij}}$ is the setup time for the jth tool used for the ith machine, and Q is the batch size. Table 15.1 shows setup time for various machine and setup time per tool as reported by Ostwald (1991).

EXAMPLE 15.2

Estimate the setup time for machining a part shown in the following figure. Note that the machining involves turning and drilling operations. The turning operation involves area turning (removing material on the exterior cylindrical surface) and front-end surface turning as shown. In addition, two drill operations are to be carried out for the hole at the front-end face—that is, a center drill and a hole drilling.

Solution

From Table 15.1, the setup times for NC lathe and multiple spindle drills are 1800 and 180 sec., respectively. The setup times for tools on the machines are 540 and 280 sec., respectively. Note that one cutter is employed for the turning operation. For hole-making, we need two drills: center and regular. Therefore, using Eq. 15.14, we have



$$T_{su} = (1800 + 180) + (540 + 280 \times 2) = 3{,}080 \text{ sec.} = 51.3 \text{ min.}$$

Table 15.1 Setup Time of Selected Machines

Machine Tool	Set-up Time, Machine (sec.)	Set-up Time, Tool (sec.)	Engaging Time (sec.)
Engine lathe	1620	720	—
Turret lathe	4300	800	9
N.C. lathe	1800	540	2
Knee and column mill	5500	—	30
Bed mill	5500	—	39
Vertical spindle mill	5500	—	30
Machining center	2500	180	8
Manual drill press	860	—	9
Power drill	860	—	9
Multi spindle drill	180	280	9
Horizontal bone	4600	—	30

15.3.2.2 Operation Time t_o

For machining operations, operation time is the duration laps from feed engagement to feed disengagement. Operation time consists of rough cutting time t_r, finish cutting time t_f, and approaching time t_a, that is,

$$t_o = t_r + t_f + t_a \tag{15.15}$$

Note that the operation time can be obtained by carrying out virtual machining simulation, as discussed in Chapter 11. In this subsection, we assume that the virtual machining simulation has not been performed, and that designers are seeking a quick estimate on the machining cost of the part without creating a machining simulation.

As a general approach, rough cutting time is proportional to the machining volume removed from all types of features. Rough cutting time can be calculated as

$$t_r = V_r/\text{MRR} \tag{15.16}$$

where t_r is the rough cutting time (min.), V_r is the material volume removed (in.^{3}), and MRR is the material removal rate (in.^{3}/min.). Note that MRR depends on workpiece material, cutting tool material, cutting tool geometry, and machining process. Table 15.2 lists MRR of selective materials for rough cut using the HSS tool (MDH, 1980). The material volume removed V_r for standard features, including rotational (turning), prismatic (milling), and revolving (drilling) is summarized in Appendix 15A.

The finish operation is carried out after rough cutting. This operation is very much associated with the dimensional accuracy and surface finish. Thus, finishing operation time t_f is proportional to the area of the finish cut; that is,

$$t_f = A_f/R_{sg} \tag{15.17}$$

where A_f is the finish cutting area (in.^{2}) and the R_{sg} is the surface generation rate (in.^{2}/min.), which varies from one cutting material to another, as suggested in Table 15.3 (MDH, 1980). The finish cutting

Table 15.2 MRR of Selective Materials for Rough Cut Using HSS Tool

Material	Hardness (BHN)	MRR Turning (in^3/min)	MRR Face Milling and Slot End Milling (in^3/min)	MRR Peripheral End Milling and Slab Milling (in^3/min)	MRR† Drilling (in^3/min)
Low-carbon steel	150–200	0.48	0.25 * n	0.08 * n * l	3.8
Medium-carbon steel	200–250	0.42	0.23 * n	0.05 * n * l	3.4
Alloy steel	150–200	0.50	0.22 * n	0.05 * n * l	3.6
Stainless steel	135–185	0.50	0.23 * n	0.06 * n*l	2.3
Tool steel	200–250	0.27	0.11 * n	0.02 * n * l	1.2

†*1 in. diameter tool;* n *is the number of teeth per cutter, and* l *is the cutter length in contact with the workpiece.*

Table 15.3 Suggested Surface Generation Rate

Material	Hardness (BHN)	R_{sg}, Turning (in^2/min)	R_{sg}, Face Milling (in^2/min)	R_{sg}, Peripheral End Milling (in^2/min)	R_{sg}† Reaming (in^2/min)
Low-carbon steel	150–200	13.4	3.7 * n	2.4 * n * l	8.0
Medium-carbon steel	200–250	10.9	1.7 * n	1.4 * n * l	9.1
Alloy steel	150–200	14.2	3.5 *n	1.4 * n * l	9.8
Stainless steel	135–185	7.6	3.5 * n	1.6 * n * l	5.6
Tool steel	200–250	7.7	1.7 * n	0.6 * n * l	3.7

†*1 in. diameter tool;* n *is the number of teeth per cutter, and* l *is the cutter length in contact with the workpiece.*

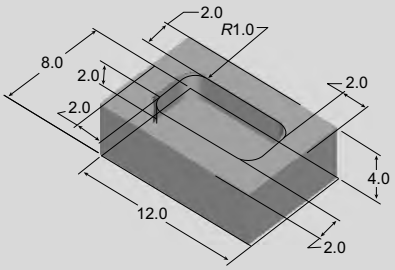
area A_f for standard features, including rotational (turning), prismatic (milling), and revolving (drilling) is summarized in Appendix 15A.

EXAMPLE 15.3

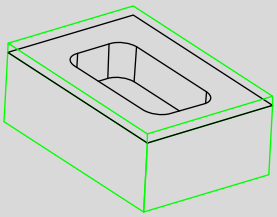
A solid block of 12 in. × 8 in. × 4 in. with a center pocket, as shown in the following figure (see next page), is to be machined from a workpiece of 12 in. × 8 in. × 4.5 in. The workpiece material is low-carbon steel. The pocket size is 8 in. × 4in. ×2 in. with fillets of radius 1 in. at four corners. Two NC sequences are used for the rough cut, a face milling that removes the layer of 0.5 in. material on the top face, and then a pocket milling that cuts the center pocket. A 1 in. diameter cutter of four teeth is employed for both sequences. A ¼ in. cutter of length 2.5 in. with four teeth is then used for finish cutting. Calculate the cutting time for both the rough and finish cuts.

Continued

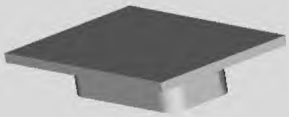
EXAMPLE 15.3—cont'd



The solid block with a pocket



Block assembled to the workpiece



Material to remove

Solution

For both face and pocket millings, the MRR shown in Table 15.2 for low-carbon steel is 0.25 n; thus, MRR = 0.25 (4) = 1 in.^{3}/min. The volume to remove for the face and pocket millings are area × thickness = (12 × 8) × 0.5 = 48 in.^{3} and 31.1 × 2 = 62.2 in.^{3}, respectively. Then, the cutting time can be calculated as

$$t_r = V_r/\text{MRR} = (48 + 62.2)/1 = 112.2 \text{ min.}$$

For the finish cut, we assume the bottom face of the pocket is a slot-end milling and the wall face is peripheral-end milling. From Table 15.3, the surface generation rates for the two sequences are 3.7 n and 2.4 nl, respectively. Thus, the finish cutting time is

$$t_f = A_f R_{sg} = 31.1/(3.7 \times 4) + (30.3)/(2.4 \times 4 \times 2.0) = 2.10 + 1.58 = 3.68 \text{ min.}$$

Note that although the cutter length is 2.5 in., the depth of the pocket is 2 in.; the length that the cutter is in contact with the pocket wall surface is 2 in. Therefore, $l = 2.0$ in., instead of the cutter length, is assumed in the calculation.

Approach time from home position to cutting position varies with the process, size of the workpiece, and location of the features to cut. In general, although approach time is measured when the feedrate is turned on, it is much less than the actual cutting time, especially for milling. As an example, empirically, the approach time (Ostwald, 1991) can be estimated as:

$$t_a = \begin{cases} 5.4 \text{ sec}, & \text{if } D > 2\text{in.} \\ 3.8\sqrt{D} \text{ sec}, & \text{otherwise} \end{cases} \tag{15.18}$$

for turning, where D is the diameter of the round bar.

The total operation time t_o for a part with k features that require m cutters to cut can be calculated as

$$t_o = t_c + \sum_{j=1}^{m} t_{a_j} \tag{15.19}$$

where t_c is the total cutting time when the cutter is in contact with the workpiece; that is,

$$t_c = \sum_{i=1}^{k} (t_{r_i} + t_{f_i}) \tag{15.20}$$

15.3.2.3 Non-operation Time t_{no}

The non-operation time in general consists of workpiece handling time t_h and tool engaging time t_e; that is,

$$t_{no} = t_h + t_e \tag{15.21}$$

The handling time includes time for handling, loading, unloading, clamping, unclamping, chip cleaning, and so on. The loading time is approximately proportional to clamping, as proposed by Boothroyd and Reynolds, (1988); that is,

$$t_h = n_h(38 + 1.1\,W) \tag{15.22}$$

where t_h is in seconds, n_h is the number of loadings/unloadings, and W is the weight of the workpiece in lb.

Tool engaging time includes time for tool position, feed engage/disengage, and feed/speed adjustment. The tool engaging times for selected machines are listed in Table 15.1 (Ostwald, 1991).

15.3.2.4 Tooling Cost C_t

Tooling cost C_t for machining operations include costs of cutters, fixtures, jigs, and so on, that are required to perform the machining task. Here we only discuss the cutter cost, which is related to cutting speed V, cutter material, and workpiece material, defined as

$$C_t = C_{\text{tool}}\frac{t_c}{T_t} \tag{15.23}$$

where C_t is the unit cost ($/unit) of cutting tool; t_c is the cutting time (min.), including rough cut t_r and finish cut t_f; T_t is the tool life (min.); and C_{tool} is the cutter cost ($). Note that in general more than one tool is used to machine a feature. Often, a number of cutters are loaded onto the tool turret to machine a single part in a batch. Therefore, 15.23 must be calculated for individual tools with the total cutting time that the tool is in contact with the workpiece. The tool life T_t can be estimated using the famous Taylor's cutting speed equation,

$$VT_t^s = K \tag{15.24}$$

where T_t is the tool life (min. per cutting edge), V is the cutting speed (in./min.), and s and K are empirical constants from field studies. The s and K of HSS and tungsten carbide tools for selected workpiece materials are listed in Table 15.4 (Ostwald and McLaren, 2004).

Table 15.4 Taylor's Tool Life Parameters

Tool Material	HSS		Tungsten Carbide	
Work material	K	s	K	s
Stainless steel	170	0.08	400	0.16
Medium-carbon steel	190	0.11	450	0.20
Gray cast steel	225	0.12	3000	0.43

EXAMPLE 15.4

For a HSS cutter that is used to machine a batch of gray cast steel workpieces, if the cutting speed is assumed to be $V = 100$ in./min., calculate the cutter life of the HSS cutter.

From Table 15.4, Taylor's empirical parameters s and K for the HSS cutter machining a gray cast steel workpiece are $s = 0.12$ and $K = 225$, respectively. Thus, using 15.24 the tool life can be estimated as

$$T_t = (K/V)^{1/s} = (225/100)^{1/0.12} = 860 \text{ min.}$$

Note that if the cutting speed increases to 150 in./min., the tool life becomes

$$T_t = (225/150)^{1/0.12} = 29.3 \text{ min.}$$

A higher cutting speed could significantly reduce the tool life of the cutter based on the Taylor's equation.

15.3.2.5 Material Cost C_{mat}

Often, the most important factor in the total cost of a machined component is the workpiece's material cost. This frequently forms more than 50% of the total costs and, therefore, should be estimated with reasonable care. In general, the material cost for a machined part can be calculated using Eq. 15.2; that is,

$$C_{\text{mat}} = V_w \, \rho \, R_{\text{mat}} \tag{15.2}$$

where V_w is the volume of the workpiece, ρ is the weight density of the workpiece material, and R_{mat} is the material cost per unit weight.

EXAMPLE 15.5

Estimate the material cost of the workpiece of Example 15.3. The size and material of the workpiece are 12 in.$\times$ 8 in. $\times$ 4.5 in. and low-carbon steel, respectively. The weight density of low-carbon steel is 0.284 lb./in.^{3} The cost of steel is about \$3.35/lb.

Solution

From Eq. 15.2, the material cost can be estimated as

$$C_{\text{mat}} = V_w \, \rho \, R_{\text{mat}} = (12 \times 8 \times 4.5) \times 0.284 \times 3.35 = \$411$$

15.3.3 INJECTION MOLDING COST MODEL

Injection molded products appear in every sector of product design: consumer products, business, industrial, computers, communication, medical devices, toys, cosmetics packaging, and sports equipment. The most common equipment for molding thermoplastics is the reciprocating screw machine (see, for example, Kalpakjian and Schmid, 2010 or Boothroyd et al., 2011). Polymer granules are fed into a spiral press where they mix and soften to a dough-like consistency that can be forced into the die through one or more channels (including sprue and runners). The polymer solidifies under pressure and the component is then ejected.

Injection molding requires relatively expensive tooling. For this reason, injection molding is suited for large production volumes. The rate for production can be high, particularly for small moldings. Multi-cavity molds are often used. The overall cost of an injection molded part is driven mainly by the materials cost, the required equipment, the cycle time on the machine, and the required (and dedicated) tooling.

Like that of Eq. 15.8, the cost of an injection molded part can be calculated as

$$C_u = C_p + C_t + C_{\text{mat}} = R_p t_{\text{unit}} + (C_n + C_b)/Q + C_{\text{mat}} \tag{15.25}$$

where R_p is the sum of cost rates of the injection machine rate, R_m, and direct labor rate, R_ℓ, with overhead (see Eq. 15.5); C_n and C_b are the costs of the mold and mold base, respectively; and C_{mat} is the material cost per component. For injection molding, manufacturing time per unit depends on cycle time t_i and the number of cavities n_c designed in the molds; that is,

$$t_{\text{unit}} = t_i/n_c = (T_{su}/Q + t_o + t_{no})/n_c \tag{15.26}$$

Note that n_c represents the number of cavities in the mold (i.e., number of parts produced per injection cycle).

15.3.3.1 Machine Rate R_m

As shown in Eq. 15.10, the machine rate R_m depends on equipment purchased cost C_e, the years of payback period Y_p, and the number of work shifts N_s. A review of the capital costs of injection molding machines of a principal U.S. manufacturer shows that these can be directly associated with available clamping force. A plot of machine cost versus clamping force shows a linear relationship between machine cost and clamping force (Dewhurst and Boothroyd, 1988), which can be represented by

$$C_e = 15{,}890 + 42.86\, F_c \tag{15.27}$$

where C_e is injection molding machine cost in \$ and F_c is the machine clamping force in kN. The clamping force is required to clamp the molds while injecting materials. Note that Eq. 15.27 was published a long time ago. Therefore, inflation must be taken into consideration before using the equation. Once a machine cost is obtained based on the calculated clamping force, the machine rate can be estimated using Eq. 15.10 with more input such as years of payback period and the number of work shifts.

Another approach for estimating the clamping force (and machine rate) proposed by Boothroyd et al. (2011) is more straightforward. Boothroyd et al. compiled a list of material data commonly employed for injection molding and the respective injection pressure required, as shown in Table 15.5. The required clamping force can be calculated by multiplying the pressure with the projected cavity area (plus sprue and runners). Moreover, Boothroyd et al. (2011) suggest that, as a general rule, approximately 50% of the pressure generated in the machine is lost as a result of the flow resistance in the sprue, runner systems, and gates. Therefore, only 50% of the pressure is used for calculating the clamping force; that is:

$$F_c = 50\%\, PA \tag{15.28}$$

where F_c is the clamping force, P is the injection pressure, and A is the projected area of cavities plus sprue and runners.

With the clamping force calculated, Boothroyd et al. (2011) provide a selection of injection molding machines driven by the clamping force, as shown in Table 15.6, in which machine rate (under the "operating cost" column) is provided, among other important machine data.

Table 15.5 Material Data for Injection Molding

Thermoplastic	Specific Gravity (g/cm^3)	Thermal Diffusivity (mm^2/s)	Injection Temp (°C)	Mold Temp (°C)	Ejection Temp (°C)	Injection Pressure (bars)
High-density polyethylene	0.95	0.11	232	27	52	965
High-impact polystyrene	1.59	0.09	218	27	77	965
Acrylonitrile-butadien-styrene (ABS)	1.05	0.13	260	54	82	1000
Acetal (homopolymer)	1.42	0.09	216	93	129	1172
Polyamide (6/6 nylon)	1.13	0.1	291	91	129	1103
Polycarbonate	1.2	0.13	302	91	127	1172
Polycarbonate (30% glass)	1.43	0.13	329	102	141	1310
Modified polyphenylene oxide (PPO)	1.06	0.12	232	82	102	1034
Modified PPO (30% glass)	1.27	0.14	232	91	121	1034
Polypropylene (40% talc)	1.22	0.08	218	38	88	965
Polyester terephtalate (30% glass)	1.56	0.17	293	104	143	1172

Boothroyd et al., 2011

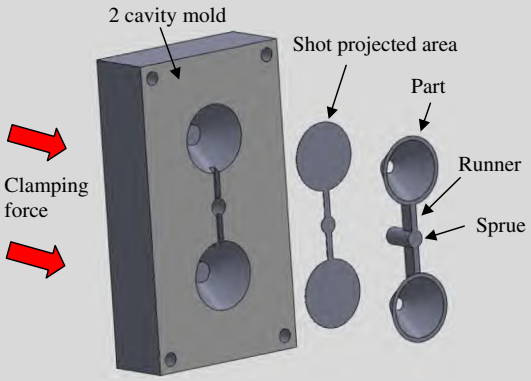
Table 15.6 Selective Injection Machines

Clamping Force (kN)	Shot Size (cc)	Operating Cost[a] ($/h)	Dry Cycle Times (s)	Maximum Clamp Stroke (cm)	Driving Power (kW)
300	34	28	1.7	20	5.5
500	85	30	1.9	23	7.5
800	201	33	3.3	32	18.5
1100	286	36	3.9	37	22.0
1600	286	41	3.6	42	22.0
5000	2290	74	6.1	70	63.0
8500	3636	108	8.6	85	90.0

[a]*Machine rates will vary by country, region, and year—text values are intended for design comparisons only.*

EXAMPLE 15.6

A batch of thin-shell cone-shaped parts of thickness 5 mm and base diameter 10 cm are to be molded from acrylonitrile-butadiene-styrene (ABS) in a two-cavity mold, similar to the one shown in the following figure. Assuming that the projected area due to the runner system is 10% of the projected cavity area, calculate the clamping force and select an adequate machine and machine rate.



Solution

The projected area of each part is $(\pi/4)(10^2) = 78.6\ \text{cm}^2$. The overall project area is then $A = (2 \times 78.6) \times 1.1 = 173.0\ \text{cm}^2$. From Table 15.5, the recommended injection pressure for ABS is 1000 bars (1 bar is 0.1 MPa = 100,000 Pa). Therefore, the clamping force required is

$$F_c = 50\%\ \text{P A} = 50\%\left(173.0 \times 10^{-4}\text{m}^2\right)\left(1000\ \text{bars} \times 100{,}000\ \text{N}/\text{m}^2\right) = 865\ \text{kN}$$

With the available machines listed in Table 15.6, the machine would be the one with a maximum clamping force of 1100 kN, for which the machine rate is $36/hr.

15.3.3.2 Molding Cycle Time t_o

The molding cycle time on the machine consists of the injection time, which depends on the shot size and machine power, but is usually small (1–2 sec.); and the cooling time—usually dominates—as shown in Figure 15.13.

The cooling time, which is largely driven by the square of the maximal wall thickness, can be approximated as follows:

$$t_c = \frac{h_{\text{max}}^2}{\pi^2 \alpha} \log_e \frac{4(T_i - T_m)}{\pi(T_x - T_m)} \tag{15.29}$$

where

h_{max} = maximum wall thickness of part, mm
T_x = recommended part ejection temperature, °C
T_m = recommended mold temperature, °C

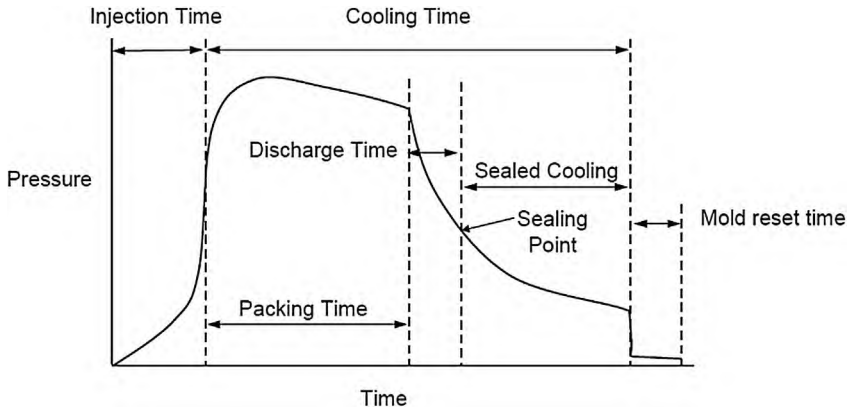


FIGURE 15.13

Injection molding cycle (Boothroyd et al., 2011).

T_i = polymer injection temperature, °C
α = thermal diffusivity coefficient, mm^2/s
As suggested by (Lovejoy et al., 2010), as a rough estimate, the cycle time can be simplified as

$$t_o = \frac{h_{\max}^2}{\alpha} \tag{15.30}$$

15.3.3.3 Mold Cost Estimation

The mold cost consists of the cost of fabricating its base—the required plate, pillars, guide bushings, and so on—and the cost of cavity and core plate fabrication. We will approximate these costs using simplified versions of the logic proposed by Boothroyd et al. (2011).

From a survey of fabricated mold bases, it has been shown by Dewhurst and Kuppurajan (1987) that mold base cost depends on the surface area of the base plates and the combined thickness of cavity and core plates. To get these numbers, you can draw your part within a rectangle as if you are looking down on the tool, seeing its length and width but not depth. That is, sketch the outlines of the parts and draw a rectangle around it (or them, if making multiple parts at one shot). Leave 2 in. between any parts and between the part and the edge of the rectangle. This 2 in. leeway is to allow room for runners, cooling channels, and other things that a tool requires. For example, the base for the mold shown in Example 15.6; there are two parts in one shot and you will have two cavities, all of which are at least 2 in. from each other and 2 in. from the edge of the circumscribing rectangle.

Dewhurst and Kuppurajan suggested the following equation for estimating the mold base cost C_b ($),

$$C_b = 1000 + 0.45\, A_c\, h_p^{0.4} \tag{15.31}$$

where A_c is the area of the mold base cavity plate (cm^2)—that is, the area of the square enclosing the parts, including a 2 in. boundary on all sides. Also, h_p is the combined thickness of cavity and

core plates in the mold base (cm), which comes from measuring the depth of the actual part because it will lie within the box. There is no need to add a boundary to this, just measure the part itself.

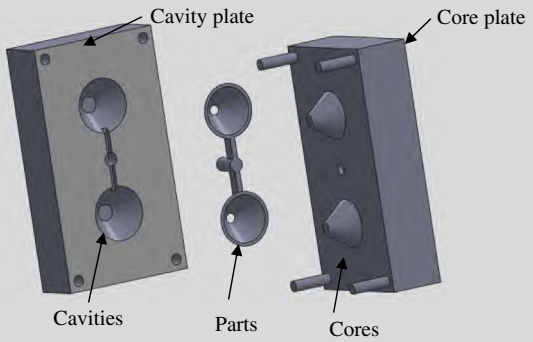
EXAMPLE 15.7

Estimate the mold base cost for the injection molding shown in the following figure. The two-cavity mold, consisting of cavity and core plates, is molding a batch of thin-shell cone-shaped parts of thickness 5 mm and base diameter 10 cm; this is similar to the one of Example 15.6. The depth of the cone is 5 cm and the length, width, and height of the cavity plate are 25 cm × 45 cm × 10 cm. The thickness of the core plate is 10 cm. Approximate the cost of the mold base.

Solution

The area of the mold base cavity plate A_c and the combined thickness h_p are $A_c = 25 \text{ cm} \times 45 \text{ cm} = 1125 \text{ cm}^2$ and $h_p = 5 \text{ cm} + 10 \text{ cm} = 15 \text{ cm}$, respectively. Thus, the cost of the mold base can be approximated using Eq. 15.31 as

$$C_b = 1000 + 0.45\, A_c\, h_p^{0.4} = 1000 + 0.45(1{,}125)(15)^{0.4} = \$2500$$



In general, manufacturing the mold is carried out almost entirely by machining. Therefore, cost for the mold C_{mold}, consisting of cavity and core plates, can be estimated by using Eq. 15.12.

In Eq. 15.12, the operation time t_o consists of machining hours for the ejection system and the mold. Boothroyd et al. (2011) cite a number of studies that estimate mold costs as a function of various inputs. For example, the number of ejector pins needed in a tool is roughly equal to the square root of the projected part area (cm^2) (i.e., $N_e = A_p^{0.5}$). A rough estimate is 2.5 manufacturing hours per ejector pin for the ejection system (i.e., $t_{es} = 2.5\, A_p^{0.5}$).

Empirically, the complexity of the part is determined by its number of "surface patches", which are continuous surfaces that a tool can traverse without retracting. For example, the mold of Example 15.6 requires a minimum of three retracts (i.e., after machining the first cavity and the runner) plus a finish cut. The machining hours can be estimated as $t_{ep} = 900 \times (0.08 + 0.02\, S_P)^{1.27}$, where S_P is the number of surface patches.

In addition to its complexity, the size of the mold affects machining costs by adding more hours, again empirically as $t_{ms} = 100 + 40\, A_p^{1.2}$, to the machining hours. So, in summary, the total number of hours to manufacture the mold is

$$t_o = t_{es} + t_{ep} + t_{ms} = 2.5 A_p^{0.5} + 900 \times (0.08 + 0.02\, S_P)^{1.27} + 100 + 40\, A_p^{1.2} \tag{15.32}$$

15.3.3.4 Material Cost C_{mat}

Material cost per part C_{mat} can be calculated as

$$C_{\text{mat}} = \frac{V_m \rho R_{\text{mat}}}{1 - f} / n_c \tag{15.33}$$

where V_m is the volume of the injection material per shot, ρ is the weight density of the material, R_{mat} is the material cost rate (\$/lb.), f is the percentage of runner and sprue to volume V_m, and n_c is the number of cavities.

15.3.4 SHEET METAL STAMPING COST MODEL

Stamping is a process that can be applied to most ductile metals. Unless employing a progressive die, you will have to cut the blanks before stamping the parts. Therefore, most stamping processes are comprised of two steps, blanking and stamping, as illustrated in Figure 15.14. In the blanking step, a shearing operation separates individual blanks of appropriate size from either larger blanks or coils. The second step, stamping, is the actual forming operation in which the part is drawn in one or several operations (called "hits"). The costs for sheet metal stamping include processing cost, tooling costs, and material cost, similar to that of Eq. 15.8.

15.3.4.1 Material Cost C_{mat}

The material costs for the blanking step are the costs per blank. Usually the blank is larger than the final part and the portion cut off during or after stamping is called an engineered scrap (or manufactured scrap). Clever part layout on the coil can reduce scrap. The exact blank size is difficult to determine, but at a minimum it should include the part surface plus some material to control the flow of material during drawing—as a rule of thumb a minimum of 2 in. on all sides—plus extra material for irregular shapes. The material that is not used for the part will receive scrap credit. Therefore, material cost per unit can be defined as

$$C_{\text{mat}} = [(W_b R_b - (W_b - Q\, W_p) R_s]/Q \tag{15.34}$$

where W_b and W_p are the weight of the sheet per press and weight of the part, respectively; R_b and R_s are the cost per weight of the blank and scrap, respectively; and Q is the number of parts per blanking operation.

A simpler way for approximating material cost is by assuming 10% engineered scrap. With this assumption, the total material cost, including scrap, is the material cost of the part multiplied by 1.1. For a set of common material prices, see Table 15.7 (Boothroyd et al., 2011).

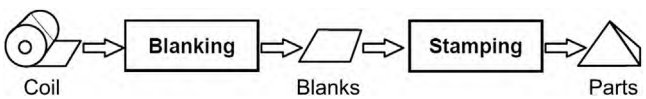


FIGURE 15.14

Basic two-step stamping process.

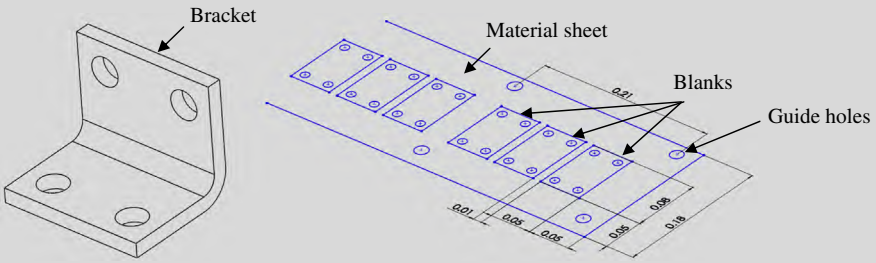
Table 15.7 Sheet Metal Properties and Typical Costs

Alloy	Cost[a] ($/kg)	Scrap[a] Value ($/kg)	Specific Gravity (g/cc)	S_{ut} (MN/m^2)	Elastic Modulus (GN/m^2)	Maximum Tensile Strain
Steel, low-carbon commercial quality	0.80	0.09	7.90	330	207	0.22
Steel, low-carbon, drawing quality	0.90	0.09	7.90	310	207	0.24
Stainless steel T304	6.60	0.40	7.90	515	200	0.40
Aluminum, 1100, soft	3.00	0.80	2.70	90	69	0.32
Aluminum, 1100, half hard	3.00	0.80	2.70	110	69	0.27
Aluminum, 3003, hard	3.00	0.80	2.70	221	69	0.02
Copper, soft	9.90	1.90	8.90	234	129	0.45
Copper, 1/4 hard	9.90	1.90	8.90	276	129	0.20
Titanium, Grade 2	19.80	2.46	4.50	345	127	0.20
Titanium, Grade 4	19.80	2.46	4.50	552	127	0.15

[a]*Material costs are subjected to constant change and are strongly influenced by the quantity of purchase—text values are intended for design comparison exercises only.*

EXAMPLE 15.8

A bracket of 0.01 m thickness shown in the following figure is manufactured by stamping. In the blanking step, aluminum sheet (AL3003) is cut into three part blanks per press as illustrated. Note that the size of the blank is 0.08 m × 0.05 m and the diameter of the holes is 0.01 m. The average sheet size processed per press is 0.18 m × 0.21 m, as shown in the dimensions in the figure. Calculate material cost per unit.



Solution

From Table 15.7, the material cost and scrap rates for AL3003 are R_b = \$3.00/kg and R_s = \$0.80/kg, respectively. The weight of the sheet per press is $W_b = V_b\,\rho = (0.18\text{ m} \times 0.21\text{ m} \times 0.01\text{ m}) \times 2700\text{ kg/m}^3 = 1.02$ kg, and the part weight is $W_p = V_p\,\rho = \{[0.08\text{ m} \times 0.05\text{ m} - \pi(0.01)^2] \times 0.01\text{ m}\} \times 2700\text{ kg/m}^3 = 0.0995$ kg. And the number of parts per blanking operation is $Q = 3$. From Eq. 15.34,

$$C_{\text{mat}} = [(W_bR_b - (W_b - Q\,W_p)R_s]/Q = [(1.02 \times 3 - (1.02 - 3 \times 0.0995)0.80]/3 = \$0.772$$

15.3.4.2 Processing Cost Rate R_p

As shown in Eq. 15.9, the processing cost, C_p, can be obtained by multiplying the processing rate, R_p, and processing time per unit t_{unit}. As also stated in Eq. 15.14, the processing rate R_p (\$/hr.) is the sum of cost rates of the machine, R_m, and direct labor, R_ℓ, with overhead. As mentioned before, in general, the machine rate R_m depends on equipment purchased cost C_e, the years of payback period Y_p, and the number of work shifts N_s.

The machine costs for the blanking process are determined by key parameters, including bed size of the machine (must be able to accommodate the blank size), press force, and press stroke. Table 15.8 lists some example machines. The press force must support the required shearing force, which depends on the material type, the material thickness, and the length of the cut. As a rule of thumb, the following equation can be used to approximate the required press force:

$$F_p = 0.5\ S_{ut}\, h\, L_s \tag{15.35}$$

where F_p is the required press force in kN, S_{ut} is the ultimate tensile strength, h is material thickness in meters, and L_s is the length to be sheared in meters. Once minimum machine size has been identified, hourly costs and run rates determine the machine cost per part for the blanking process.

EXAMPLE 15.9

Choose a machine and report the machine rate to cut the blanks of the bracket in Example 15.8.

Solution

The length to be shear is $L_s = 3[2(0.08 + 0.05) + 4\pi(0.01)] = 1.16$ m. Thickness of the aluminum sheet is $h = 0.01$ m. From Table 15.7, the ultimate tensile strength of Aluminum 3003 is $S_{ut} = 221$ MPa. Thus, from Eq. 15.35, the required press force is

$$F_p = 0.5\ S_{ut}\, h\, L_s = 0.5 \times 221{,}000{,}000 \times 0.01 \times 1.16 = 1280 \text{ kN}$$

From Table 15.8, the third press is chosen since the press force is 1,750 kN, which is greater than the required 1,280 kN. Also the width of the bed size is 150 cm, which is large enough to accommodate the sheet width of 18 cm.

Table 15.8 Mechanical Presses

Bed Size					
Width (cm)	Depth (cm)	Press Force (kN)	Operating Cost[a] (\$/h)	Maximum Press Stroke (cm)	Strokes (per min)
50	30	200	55	15	100
80	50	500	76	25	90
150	85	1750	105	36	35
180	120	3000	120	40	30
210	140	4500	130	46	15
240	175	6000	140	54	8

[a]*Machine rates will vary by country, region, and year—text values are intended for design comparisons only. If a double-action press is required for a deep drawing operation, then 20% increase should be made to the above operating costs.*

15.3.4.3 Tooling Cost

The tooling costs of the blanking process are determined by the size of the part, material type and thickness, and the shape of the cut. If the cut is a straight line, a nondedicated shear can be used. If, however, the blank has its own unique shape then a dedicated blanking tool must be purchased and amortized. For a simple cut-off, you can just add 10% to the hourly machine rate to reflect tool costs. To calculate the tool cost for blanking dies for uniquely shaped cuts, use the following formula as a lower bound:

$$C_{ds} = 120 + 0.36A_u + \left[32 + \left(\frac{P^2}{LW}\right)0.05\right] \tag{15.36}$$

where C_{ds} is the die set cost, A_u is the usable area in cm^2 between the guide pillars of the die set, P is the perimeter length to be sheared in cm, and L and W are the length and width of the smallest rectangle that surrounds the punch in cm.

The tooling costs for the stamping process depend on the part size and the number of hits. To estimate the costs of the complete tool set, you can use the following formula as an approximation:

$$C_{ts} = 1567\, N_{\text{hits}}^{1.366}\left(\frac{A_{\text{proj}}}{A_{\text{final}}}\right)^{0.323}(LW)^{0.342} \tag{15.37}$$

where C_{ts} is the tool set cost, N_{hits} is the number of hits required, A_{proj} is the projected stamping area in m^2, A_{final} is the final surface area of the part in m^2, and L and W are the length and width of the final part in mm. Stamping dies are almost always part-specific; in other words, they are dedicated and have to be amortized across the planned production volume. Stamping tools are expensive but tend to have a long lifetime as well (typically several 100,000 units).

15.3.5 ASSEMBLY COST MODEL

In an advanced manufacturing shop, product assembly lines are mostly automated. One obvious example is an automotive assembly line, where it is entirely operated by robots that drastically reduce downtime and increase productivity. On the other hand, manual assembly is still being practiced in full or in part in many industry sectors. Nowadays, some factories are still relying on manual labor for assembly lines. Whether automated or manual assembly, the operation times required in individual steps must be captured in detail for an accurate assembly cost calculation.

Due to a large variation in the nature of the assembly line, equipment and tools required, and level of skilled labor in various industries, estimating assembly cost is not straightforward to say the least. However, in general, assembly costs consist of machine and labor, which can be formulated as

$$C_a = C_p + C_t \tag{15.38}$$

where the total cost of an assembled product is the sum of the processing cost C_p and the tooling cost C_t.

Similar to part manufacturing, the processing cost C_p is an important part of unit production cost. It depends on the unit assembly time t_{unit}, and the cost rates of the machine (machine rate, \$/hr.) R_m, labor R_ℓ (labor rate, \$/hr.), and overhead. Tooling cost can vary, depending on the type of product to be assembled, among many other factors.

15.4 COMMERCIAL SOFTWARE FOR THE COST ESTIMATE

A good number of cost-estimating software tools are currently available in the commercial sector. There are CAD-based costing software that either incorporate CAD as a product data module, such as product life cycle management (PLM) systems, or costing modules that are seamlessly integrated into their respective CAD software. General-purpose costing software supports product cost estimating for a broad range of cost categories. Most such software is stand-alone but some interface with CAD. There also are special-purpose codes that focus on certain manufacturing processes; for example, some forming-simulation software offers cost estimating. In addition, several web-based costing tools are easy for novices to use.

In complying with the theme of this chapter, we focus on the cost estimate for product manufacturing. It is important to emphasize that all software employs cost templates, knowledge databases, and costing methods that are usually disclosed to end users. They provide cost estimates that are calculated following certain assumptions, and sometimes simplifications (again, mostly not disclosed to end users), and are based on inputs and parameters set by users. Nevertheless, understanding how the cost is calculated and being able to verify the accuracy of the estimated cost are critical for the user prior to accepting and using a cost-estimating tool.

In this section, a brief overview on the commercially available cost-estimating software is presented. We offer a short description for software from the individual categories mentioned previously. The strengths, weaknesses, pros, and cons of these codes will be discussed. For readers who are selecting software to use for conducting the cost estimate tasks, you may need to investigate further, request a software demo, or even acquire a short-term license for the software for hands-on evaluation.

15.4.1 CAD-BASED COSTING SOFTWARE

In recent years, major CAD companies have shifted their product development emphasis to PLM (product life cycle management)—for example, Windchill of Parametric Technologies, that developed Pro/ENGINEER (www.ptc.com/product/windchill/cost), Siemens product life cycle management (www.plm.automation.siemens.com/en_us) with UniGraphics, and PLM Solution of Dassault Systèmes (www.3ds.com/solution) that develops CATIA and commercialized SolidWorks. All these PLM software systems offer cost-estimating capabilities for product life cycle costs—generally including product manufacturing cost.

The CAD-based costing software that is of more interest is one that incorporates cost estimating as a module and seamlessly integrates it with CAD. One such software is SolidWorks Costing; its module is fully integrated and embedded in SolidWorks. The software creates automatic manufacturing cost estimates for sheet metal and machined components using built-in templates and customized data. Using SolidWorks Costing, designers can get automatic, real-time estimates of part manufacturing costs for sheet metal and machined parts. This costing module enables designers to continuously check their designs against cost targets, avoiding costly redesigns and production delays later on. Manufacturers can also use the SolidWorks cost-estimation tools to automate the quoting process. More about SolidWorks Costing will be discussed in case studies in Section 15.5.

15.4.2 GENERAL-PURPOSE COSTING SOFTWARE

Several general-purpose costing software products have been widely adopted. In this subsection, we briefly discuss some of the leading tools in this category.

15.4.2.1 SEER for Manufacturing

SEER for Manufacturing (or SEER-DFM) was developed by Galorath Co. (www.galorath.com); it focuses on manufacturing project and process options and can be used to model virtually any manufacturing operation. SEER for Manufacturing was designed to enable both intermittent and advanced users in management, finance, engineering, industrial design, and manufacturing to evaluate process options and trade-offs impacting various factors (e.g., ease of fabrication and assembly, number and availability of parts, materials selection, and failure and repair rates). Users can also optimize their process strategy by performing extensive trade-off analyses by varying assumptions and options to determine which manufacturing strategy is likely to produce the best outcome. The results of these analyses can be documented and exported to numerous third-party applications (e.g., cost reports in PDF format).

Although SEER-DFM is a powerful and highly respected cost-estimating software, it does not offer a connection to CAD systems. Users have to enter a BOM of the product manually. In addition, users must have adequate knowledge and experience in manufacturing because they have to choose adequate processes for manufacturing individual parts.

15.4.2.2 MicroEstimating

MicroEstimating, developed by Micro Estimating Systems, Inc. (www.microest.com), offers computer-aided process planning and computer-aided estimating for the machining and fabrication industries. MicroEstimating employs proprietary Machine Tool Emulation, Knowledge-Based Machining, and Automatic Feature Recognition to establish production times and costs. Equipped with libraries containing detailed machine tool specifics and material specifications, the software calculates net production times and costs with speed and precision.

One important feature of the software is its Feature Recognition technology. MicroEstimating uses SolidWorks for interpolated 3D part drawings in order to produce true net CNC machine tool production cycles and costs. The system also offers a fully integrated bill of materials application. Each bill can be nested to handle any size assembly. Importing a SolidWorks BOM is a standard system function.

15.4.2.3 DFM Concurrent Costing®

The DFM Concurrent Costing software of Boothroyd Dewhurst, Inc. (www.dfma.com/software/dfm.htm) allows users to generate accurate part and tooling cost estimates at the design concept stage. DFM Concurrent Costing software provides users with an understanding of the primary cost drivers associated with manufacturing the product—and establishes a benchmark for what the product "should cost". Central to the should-cost approach is accumulating real information about manufacturing costs and noting where specific costs are in the product design. High costs for a product are associated with its manufacture, so sharing should-cost information with suppliers makes collaborations more fruitful.

The cost models in the DFM Concurrent Costing software guide users through an assessment of alternative processes and materials, which provides cost information for the bill of materials. Costs

update automatically as users determine tolerances, surface finishes, and other part details. Gradually, as users choose effective shape-forming processes and consider how to modify part features to lower cost, the product becomes optimized.

15.4.2.4 Costimator of MTI System

Costimator® cost-estimating software (www.mtisystems.com) is designed to be easily configured to model the capabilities of a manufacturing facility. The system comes fully loaded with hundreds of process and feature-based cost models; it covers a large array of manufacturing processes and features that are implemented for a large variety of prebuilt, ready-to-use manufacturing process cost models.

Costimator employs three key methods for the product cost estimate: parametric, feature-based, and cost models. Feature-based estimating gives users with little to no manufacturing experience the ability to estimate based on the identification and selection of part features (e.g., holes, slots, bends, cut-outs) rather than manufacturing processes. Parametric cost models are developed from historical cost data. Manufacturing times and costs are generated through regression analysis, which is a capability offered within Costimator's parametric cost modeling tool.

Product cost models provide a quick and easy, yet accurate, estimating method for companies that manufacture similar products. These models combine multiple product features within a single model, enabling estimators with minimal manufacturing experience to estimate with speed, accuracy, and consistency. Based on the size of the part and its features, tool sizes and selections are performed in the background. Users only need to input items that directly refer to the part or design.

15.4.2.5 aPriori Product Cost Management

aPriori's Product Cost Management (www.apriori.com) offers capabilities that instantly determine the cost of a part or product from a CAD model, the materials to be used, and the factory where it will be produced. aPriori enables users to analyze cost in real time at the lowest level of product detail based on material type, production volume, manufacturing process, and location of manufacture.

aPriori provides support for major 3D CAD systems, enabling rapid and automatic evaluation of Geometric Cost Drivers (GCDs); that is, those aspects of the product's design that drive costs (e.g., size, shape, complexity, number of holes, number of bends, thickness, profile, tolerances, and roughness of surfaces) from the solid model. aPriori can run concurrently with the CAD application or as a stand-alone application where users simply open the CAD model from within aPriori when they are ready to perform a cost assessment. In either configuration, aPriori automatically evaluates the currently active model (part or assembly) for geometry, tolerances, surface finish, material, and relevant parameters.

GCDs are evaluated in a physics-based model of the applicable manufacturing process (e.g., machining, casting, injection molding) in combination with other (nongeometric) cost drivers such as manufacturing cost accounting methods, processes, facilities, and production parameters. aPriori determines the lowest cost manufacturing method for the part or assembly and provides that feedback to the designer in real time.

15.4.2.6 MISys Manufacturing

MISys® Manufacturing (www.misysinc.com) is designed to offer all the cost-estimating functionality a small- to medium-size manufacturing firm needs. MISys Manufacturing is a robust application designed specifically for manufacturers to gain control of all aspects of their production, from

inventory management to serial and lot tracking. The software is integrated with other popular, widely used small business accounting software.

15.4.3 SPECIAL-PURPOSE COSTING SOFTWARE

Special-purpose costing software offers cost estimates for dedicated manufacturing processes. In this category, the most common process supported is injection molding. There are several codes that support injection molding cost estimating. Injection Molding Cycle Time Estimator (imcycletimeest.sourceforge.net) is a very simple software that can be used for a rough cost estimation, including different data based on resin type; it allows machine settings to override for temperature.

ProMax-One™ software (www.injecnet.com) calculates the total cost per part and can show a breakdown window with the detail of all cost results. It considers the materials used, project information, labor, mold cost, machinery, maintenance, cavity information, and so on.

CostMate® by IDES Inc. (www.ides.com) is a molding part cost estimator that has been integrated into an online plastics search engine (Prospector®). CostMate considers costs of shipping and packaging as well.

CalcMaster® Injection Molding Software (www.selectedtechnologies.com/cm/cm.htm) is a powerful program that can be used not only as a cost estimator but also as a good design assistant. The software allows users to quickly determine mold cost, injection molding parameters, optimal number of cavities, and complete molded product cost.

15.4.4 WEB-BASED COSTING TOOLS

The most popular and useful website that supports cost estimates is probably CustomPartNet (www.custompartnet.com), which is an online resource for manufacturing cost estimation. Its tools allow users to perform quick calculations that facilitate the product design and costing process. The site also gathers the latest industry news into one location to keep users informed about the latest developments and trends in the manufacturing and product costing industries. With CustomPartNet, users are able to quickly create a new cost estimate, or find a similar part from the public parts gallery, to use as a baseline. Estimates can be saved, added to part galleries, and even shared with colleagues to collaborate on the estimation process.

CustomPartNet also provides educational content across a wide range of manufacturing processes to help both students and practicing engineers who are new to the manufacturing industry. Process overviews, design guidelines, and an in-depth glossary allow users to explore how a process works and learn how to design parts more cost effectively. The site's interactive tools include a process selector, material selector, and manufacturing widgets that perform quick calculations for common design and manufacturing problems. In addition, the website offers enough detail to explain the subtle, yet crucial, relationships between cost, part design, material performance, process selection, and supplier capabilities. At the same time, the site maintains content that is easy to understand and easy-to-use online tools.

Another website worth mentioning is Get-A-Quote.net (www.get-a-quote.net), which is an online construction cost-estimating tool for contractors. One of the useful capabilities offered is the information on material prices, which are updated regularly by the Craftsman Book Company (www.craftsman-book.com). The prices are estimates of what most contractors who buy in moderate volume will pay suppliers.

15.5 CASE STUDIES

We include three case studies here to illustrate more details of the product cost estimate for engineering applications. Cases 1 and 2 involve machining costing and sheet metal costing using SolidWorks, respectively (Sections 15.5.1 and 15.5.2). Case 3 uses SEER-DFM software for estimating the cost of a bicycle wind measurement device (Section 15.5.3).

15.5.1 MACHINING COSTING USING SOLIDWORKS

Instead of illustrating the detailed steps of using SolidWorks Costing, we offer an in-depth explanation in machining costing implemented in SolidWorks. We will also verify the calculations of cost estimates in SolidWorks and discuss the advantages and shortfalls.

We encourage readers to go over tutorial examples provided by SolidWorks for both machining costing and sheet metal costing in order to get a general idea about the costing capability and menu options. You can use the pull-down menu to access the tutorials: Help > SolidWorks Tutorials. For SolidWorks 2012 users, the costing tutorials are in Set 2, named "SolidWorks Costing", as shown in Figure 15.15(a). The two tutorial examples are shown in Figure 15.15(b).

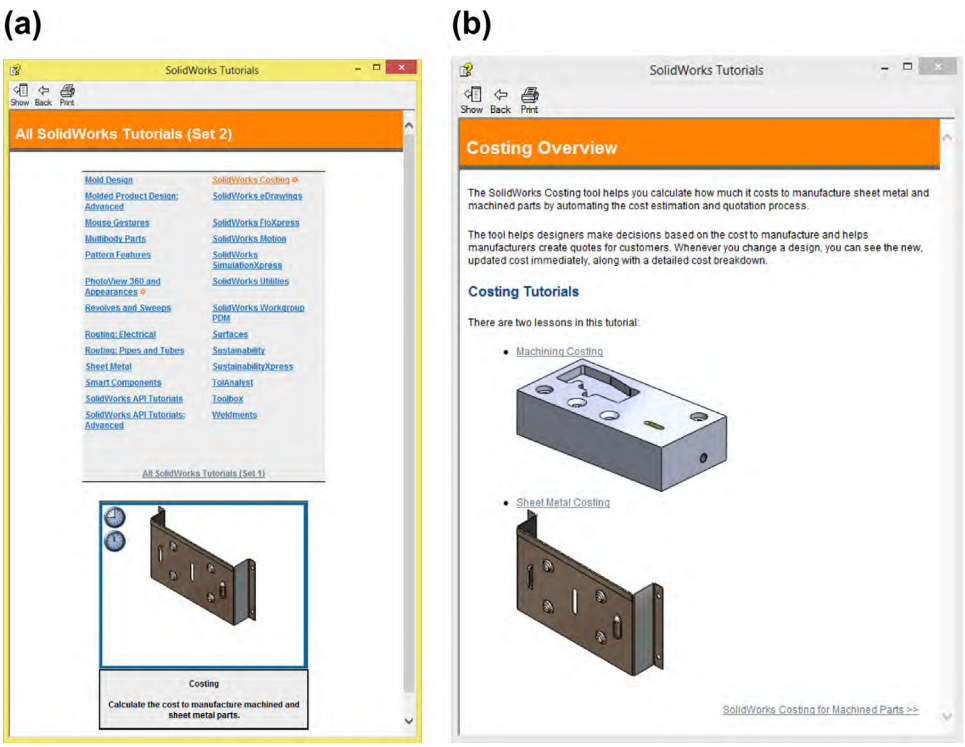


FIGURE 15.15

SolidWorks Costing tutorials: (a) Set 2 tutorials and (b) machining costing and sheet metal costing examples.

SolidWorks Costing interprets part geometry as to how it will be manufactured, not how it is designed. In the Costing tool, the solid features are interpreted as costing features in the Costing-Manager (refer to Figure 15.17(c)). For example, an extruded cut or Hole Wizard hole in SolidWorks is recognized as a drilled hole. Sometimes an entire group of SolidWorks features is recognized as one manufacturing feature in Costing; that is, the outside edges of a part might consist of fillets and straight edges. These are recognized in Costing as one cut path.

The first thing to grasp in understanding SolidWorks Costing is the cost templates. The tool provides templates that associate manufacturing features with their costs. The templates include information about material, machining, and labor costs. After recognizing the manufacturing features, Costing categorizes each feature and applies the correct information from the template to cost out the specific manufacturing features. A total cost for all the features is tabulated and a final unit cost is displayed.

The costing templates are the most relevant for machining processes, including stock material, setup cost, mill, and drill, as shown in Figure 15.16. The Stock material template stores stock

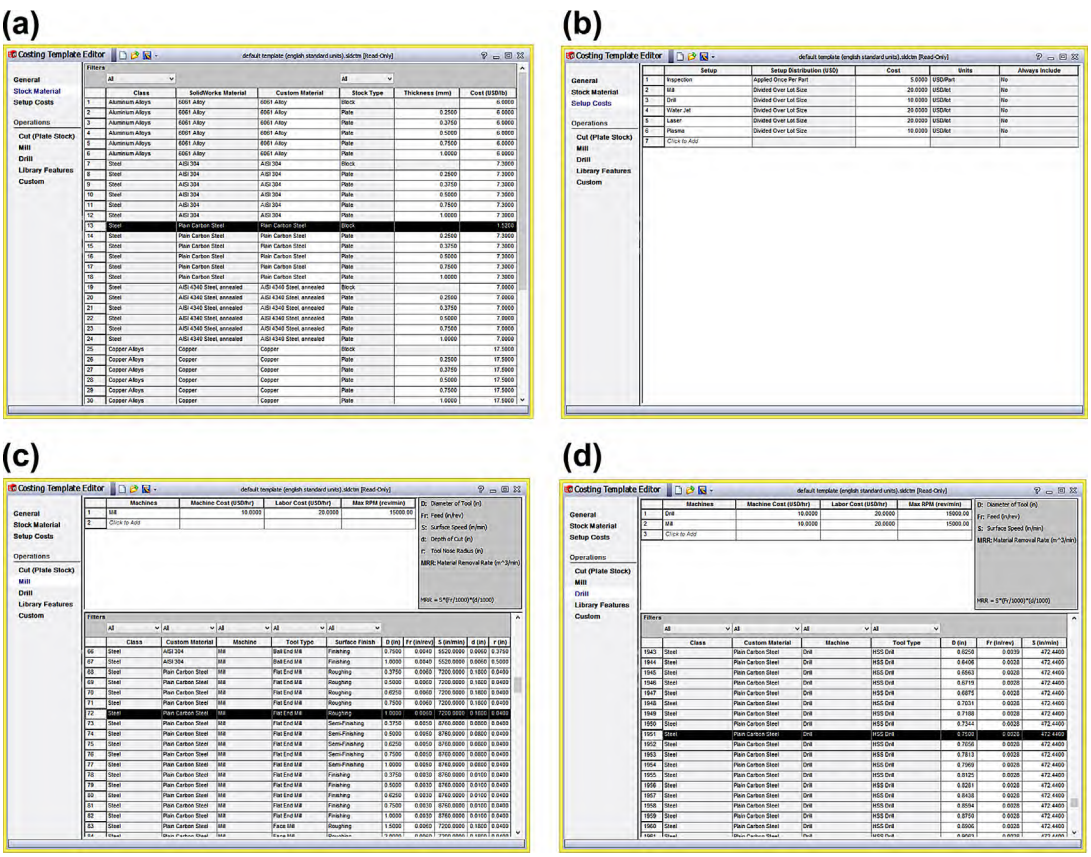


FIGURE 15.16

SolidWorks Costing templates for machining: (a) Stock material, (b) Setup cost, (c) Mill, and (d) Drill.

materials, including material type, size, and cost per weight. The Setup template allows users to define costs associated with manufacturing setups such as setting up machines to run a batch (lot) of parts. The Mill template stores the machines and costs associated with milling operations, including stock material, tool type, surface finish, tool size, feedrate, cutting speed, depth of cut, and so on. The Drill template defines the machines and costs associated with drilling operations, including stock material, tool type, feedrate, and cutting speed. Note that the costs calculated by the Costing tool are as accurate as the data stored in the templates. Although SolidWorks provides prepopulated templates, it is best to create custom templates based on your cost data.

The example we are using is a simple block of 6 in. × 4 in. × 0.75 in. with a pocket and four blind holes, both with a depth of 0.5 in., as shown in Figure 15.17(a). Note that the hole diameter is 0.75 in. and the pocket area is 6.93 in.^{2} The pocket area can be measured using the measuring tool of SolidWorks (Tools > Measure).

Accessing the Machining Costing tool is straightforward. All you have to do after you bring in the part is to choose from the pull-down menu Tools > SolidWorks Costing > Machining Costing. You can create a part like that of Figure 15.17(a) yourself or download it from the book's companion website.

After choosing Machining Costing, the Manufacturing Costing box appears to the right of the graphics window, as shown in Figure 15.17(b). Under Machining Template, choose default template (English standard unit). If you click Launch Template Editor, a template window like those of Figure 15.16 will appear. Under Materials, choose Steel for Class and Plain Carbon Steel for name.

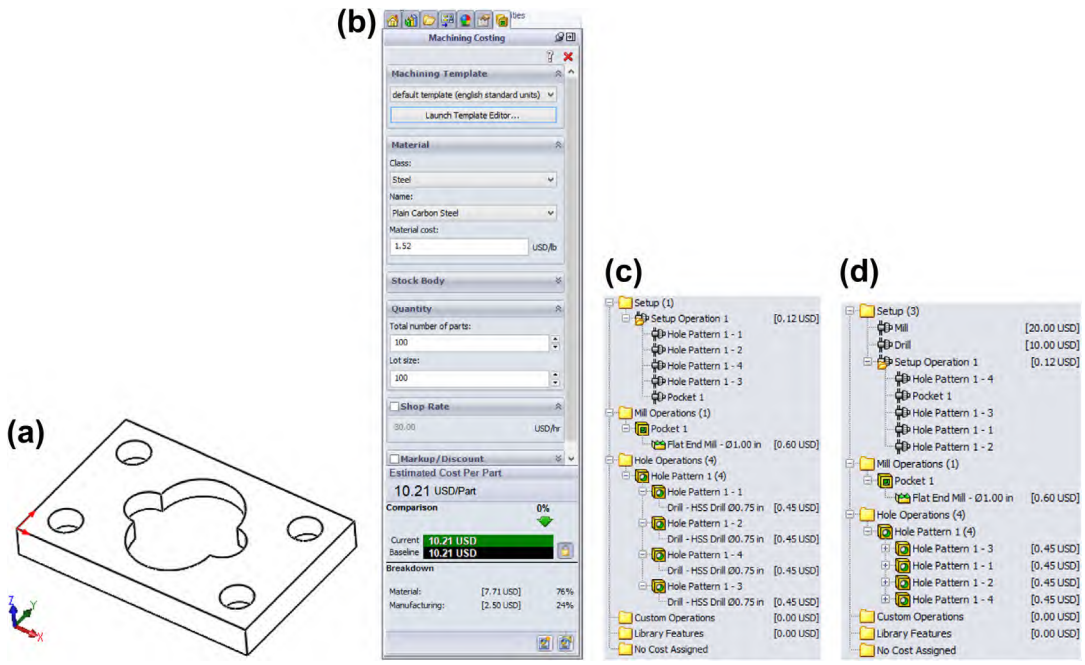


FIGURE 15.17

Defining cost information: (a) solid model, (b) Machining Costing box, (c) CostingManager, and (d) CostingManager after setup costs are added.

Under Material cost, 1.52 USD/lb. appears. By default, 100 is specified for both total number of parts and lot size. The default shop rate is $30.00/hr.

Note that the Lot size is the number of parts run in one machine setup. For example, if you run 200 parts and 100 parts per lot, you need two machine setups. For our case, we only need one machine setup. As soon as we select the material, Costing automatically calculates machining cost for the part using data stored in the default templates. As a result, the estimated cost per part is $10.21, so material cost is $7.71 (76%) and manufacturing cost is $2.5 (24%), as shown at the bottom of the Machining Costing box in Figure 15.17(b).

In the meantime, the CostingManager will appear on the left (the area of Model Tree) with manufacturing features listed, as shown in Figure 15.17(c). In the CostingManager, three major entities are listed: setup, mill operation, and hole operation. Under setup, all five manufacturing features are listed, with a cost of $0.12. Note that all cost values in the CostingManager are per part. The setup costs that are displayed do not include the setup costs for the equipment to machine the part. The cost shown currently ($0.12) is a generic 5% of the total processing cost; that is, the amount of time to remove the material for this feature multiplied by the labor and machine cost. For example, if you hover over a Hole Pattern 1-1 under Setup Operation 1, how the cost is calculated appears as 0.0223 USD = 0.05 * 0.45 USD.

Note that 0.45 USD is the cost of drilling the hole as listed under Hole Operation > Hole Pattern, Hole Pattern 1-1. We need to include the setup costs for the mill and drill to run the batch of 100 parts. To simplify the cost calculation, we set both the total number of parts and the lot size to 1 in the Machining Costing box. Note that the estimated cost per part is unchanged and so are the individual costs listed in the CostingManager.

Now we add setup costs. To manufacture this part, we need a milling machine and a drill press. These two setup costs can be added by right-clicking the Setup entity in the CostingManager and choosing Select Setup Cost, then choose Mill; do the same for the drill. The setup costs $20 and $10 are added to the CostingManager under Setup for Mill and Drill, respectively, as shown in Figure 15.17(d). Thus, the total cost becomes $40.21 after adding the two setup costs. Note that the setup cost is already stored in the machining template and recognized by the Costing tool. The total cost calculated, $40.21, is for machining one part. Is the estimate accurate? How is the total cost calculated in SolidWorks Costing?

First, the setup costs of the mill and drill are straightforward. They are defined in the Setup Costs template shown in Figure 15.16(b). As also mentioned earlier, the cost of the Setup Operation is 5% of the overall processing costs, including mill and hole operations—$0.05 \times (0.6 + 4 \times 0.45) = \$0.12$. The question is: How are the costs of mill and hole operations calculated?

For the pocket mill operation, we know from the CostManager that a flat endmill of ϕ1.00 in. is used. If we hover over the Flat EndMill ϕ1.00 in entity under Pocket 1 in the CostManager, we see the mill processing cost for the pocket is 0.5986 USD = 1.20 min. $\times$ (10.00 USD/hr. + 20.00 USD/hr.). The machine and labor rates are 10.00 USD/hr. and 20.00 USD/hr., respectively, as shown in the Mill cost template of Figure 15.16(c). Now, where does the 1.20 min. machining time come from?

From the Mill cost template, we have the following data:

Diameter of the tool $D = 1.00$ in.
Feed $Fr = 0.0060$ in./rev
Surface Speed $V = 7200.00$ in./min.
Depth of cut $d = 0.180$ in.

As a result, the spindle speed is $N = V/(\pi D) = 7200/(\pi \times 1) = 2290$ rpm. Therefore, the feedrate is $f = Fr\,N = 0.006 \times 2{,}290 = 13.8$ in./min. Thus, the material removal rate for an end mill is MRR $= D\,d\,f = 1 \times 0.18 \times 13.8 = 2.48$ in.^{3}/min. We also know that the area and depth of the pocket are 6.93 in.^{2} and 0.5 in., respectively. Thus, the volume of the pocket is $V_r = 6.93 \times 0.5 = 2.97$ in.^{3} and the machining time is $t = V_r/\text{MRR} = 2.97/2.48 = 1.20$ min.

Certainly, this is a very simple approach for cost calculation. Plus the cost is calculated for machining one pocket as roughing. For a machining time other than roughing, you can choose finish or semifinish under Default surface finish for mill operations in the Costing template by choosing the General option, as shown in Figure 15.18(a).

Similarly, for the drill operation, the cost can be calculated by first calculating the MRR for the drill. From the Drill costing template (Figure 15.16(d)), we have the following data:

Diameter of the tool $D = 0.75$ in.
Feed $Fr = 0.0028$ in./rev
Surface Speed $V = 472.44$ in./min.

Thus, the spindle speed is $N = V/(\pi D) = 472.44/(\pi \times 0.75) = 200$ rpm. Therefore, the feedrate is $f = Fr\,N = 0.0028 \times 200 = 0.560$ in./min. The material removal rate for a drill is MRR $= A\,f = ¼ (\pi \times 0.75^2) \times 0.560 = 0.247$ in.^{3}/min. We also know that the volume of the hole is $V_r = A\,d = ¼$

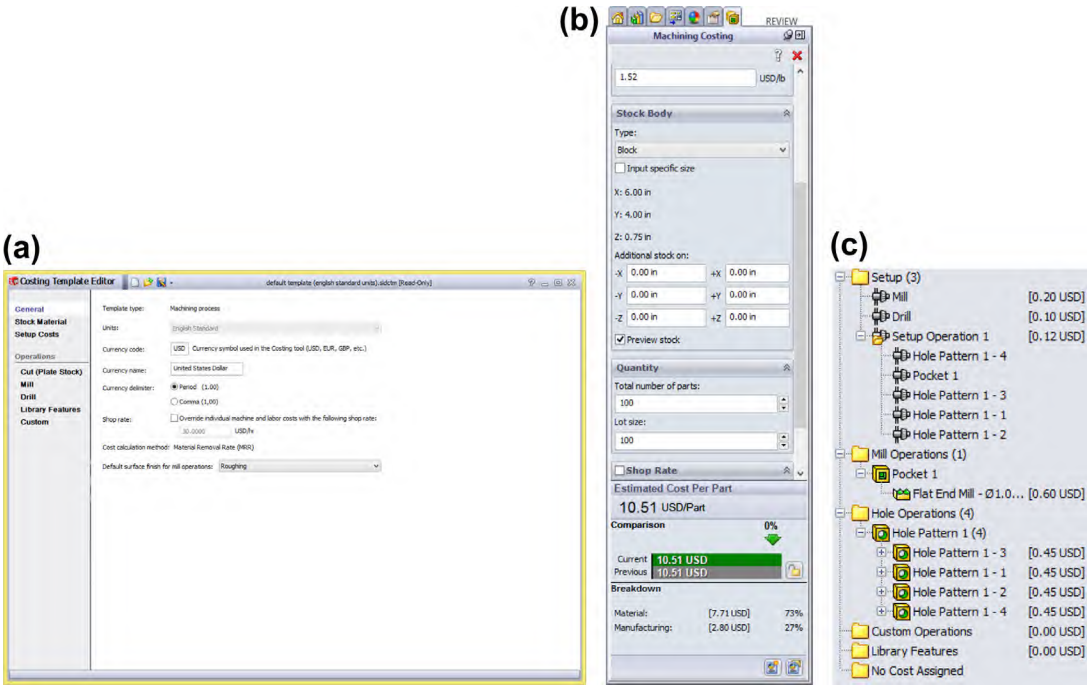


FIGURE 15.18

Cost estimate for machining: (a) Costing template, (b) cost of machining 100 parts, and (c) changes in setup costs.

$(\pi \times 0.75^2) \times 0.5 = 0.221$ in.^{3} Thus, the machining time is $t = V_r/\text{MRR} = 0.221/2.47 = 0.895$ min. $= 0.0149$ hr.; the cost of hole operation is $0.0149 \times (10.00$ USD/hr. $+ 20.00$ USD/hr.$) = 0.447$ USD.

Material cost can be calculated by the total weight of the workpiece and the price per weight. From the Costing template, the stock material for a plain carbon steel block is \$1.52/lb. The weight density of the plain carbon steel is 0.2818 lb./in.^{3} and the block volume is 6 in. $\times$ 4 in. $\times$ 0.75 in. $=$ 18.0 in.^{3} Thus, the material cost of the block is 18.0 in.^{3} $\times$ 02818 lb/in.^{3} $\times$ \$1.52/lb $=$ \$7.71.

The preceding cost estimate is based on the assumption of machining one part. If we increase the number of parts to 100, then the total per part cost becomes \$10.51, with \$7.71 and \$2.80 for material and manufacturing, respectively, as shown in Figure 15.18(b). Note that the processing costs are unchanged since they are calculated per part. The setup costs of the mill and drill become \$0.20 and \$0.10, respectively; these are obtained by dividing the setup costs (\$20 and \$10, respectively) by the number of parts (100), as shown in Figure 15.18(c).

As can be seen from this case study, the machining costing module in SolidWorks is simple and straightforward. One advantage is that the cost information is readily available for designers as the part design materializes. In addition, a number of what-if studies, such as changing part material, varying workpiece size, and varying dimension of features, can be quickly carried out. You can set the current case as the baseline one, then compare costs estimated for all the follow-on "what-if" scenarios with the baseline case to gain a clear understanding on the implication of design changes on part cost.

In all cases, overall part cost can be calculated in real time, which greatly helps the designer explore design alternatives with cost in mind. However, as mentioned before, costs calculated by the Costing tool are as accurate as the data in your templates. It is critical to check template data regularly to make sure what is there is accurate and up to date. You can consult with machine shop personnel to find more realistic figures for setups and rates, both labor and machine. One important cost that SolidWorks does not include is tool and fixture cost, among others such as overhead.

15.5.2 SHEET METAL COSTING USING SOLIDWORKS

Similar to machining, SolidWorks Costing for sheet metal interprets geometry as to how it will be cut and bent, not how it is designed. For example, in sheet metal costing, a hole through a sheet metal part is recognized as a cut path. This cut path will be manufactured using laser, waterjet, or plasma cutting.

Similar to machining costing, the first thing to grasp to understand SolidWorks Costing is the cost templates. Four costing templates are the most relevant for sheet metal cost, setup cost, thickness (with materials), cut, and bend, as shown in Figure 15.18. The Thickness template stores sheet materials, including material class, custom material, thickness (gauge and inch), and cost per weight. The Setup template allows users to define any costs associated with sheet metal setups such as setting up inspection to run a batch (lot) of parts. The Cut template stores setup costs for length cut (e.g., waterjet) and stroke cut (e.g., punch) machines and costs associated with cutting operations, including sheet material, thickness, cut method, and cost (\$/length). You can use options in the filter to show the unit cost of other cutting methods such as punch (\$/stroke). The Bend template defines the sheet material and costs associated with the bend operations.

Note that, as with the machining costing, the costs calculated by the Costing tool for sheet metal are as accurate as the data in your templates. Although SolidWorks provides prepopulated templates, it is best to create custom templates based on your cost data.

The example we are using is a thin-shell bracket of a 5.5 in. × 4 in. × 1.53 in. bounding box with a center oval cut, four holes on the flanges, and four bends, as shown in Figure 15.19(a). Note that the hole diameter is 0.375 in. and the perimeter of the center oval cut is 7.28 in. Like machining, accessing the Sheet Metal Costing tool is straightforward. All you have to do after you bring in the part is to choose Tools > SolidWorks Costing > Sheet Metal Costing. You can download the sample part shown in Figure 15.19(a) from the book's companion website.

After choosing Sheet Metal Costing, the Sheet Metal Costing box appears to the right of the graphics window, as shown in Figure 15.19(b). Under Costing Template, choose default template (English standard unit). If you click Launch Template Editor, a template window like those of Figure 15.20 will appear. Under Materials, choose Steel for Class and Plain Carbon Steel for name. Under Thickness, 21 gauge (0.0329 in) appears from template and 1.52 USD/lb. appears under Material cost. By default, 100 is specified for the total number of parts and lot size. As soon as we select the material, Costing automatically calculates sheet metal cost for the part using data stored in

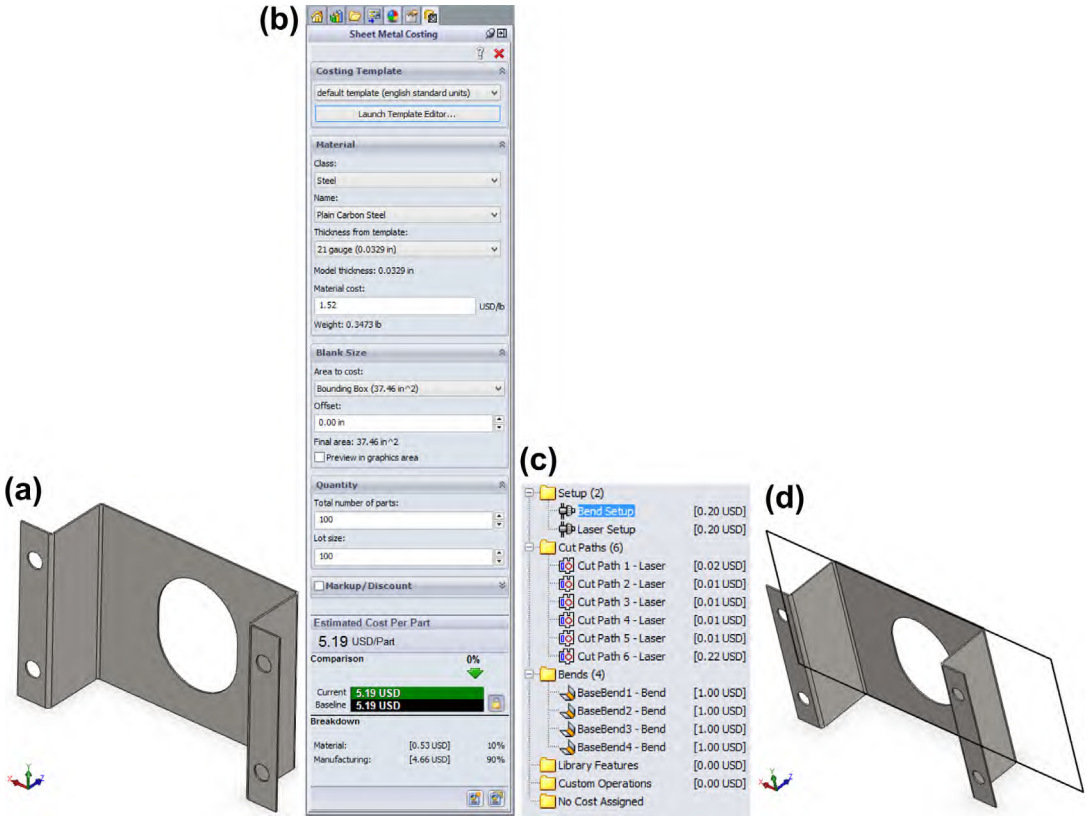


FIGURE 15.19

Defining cost information: (a) the bracket thin-shell model, (b) Sheet Metal Costing box, (c) CostingManager, and (d) bounding rectangle of the bracket.

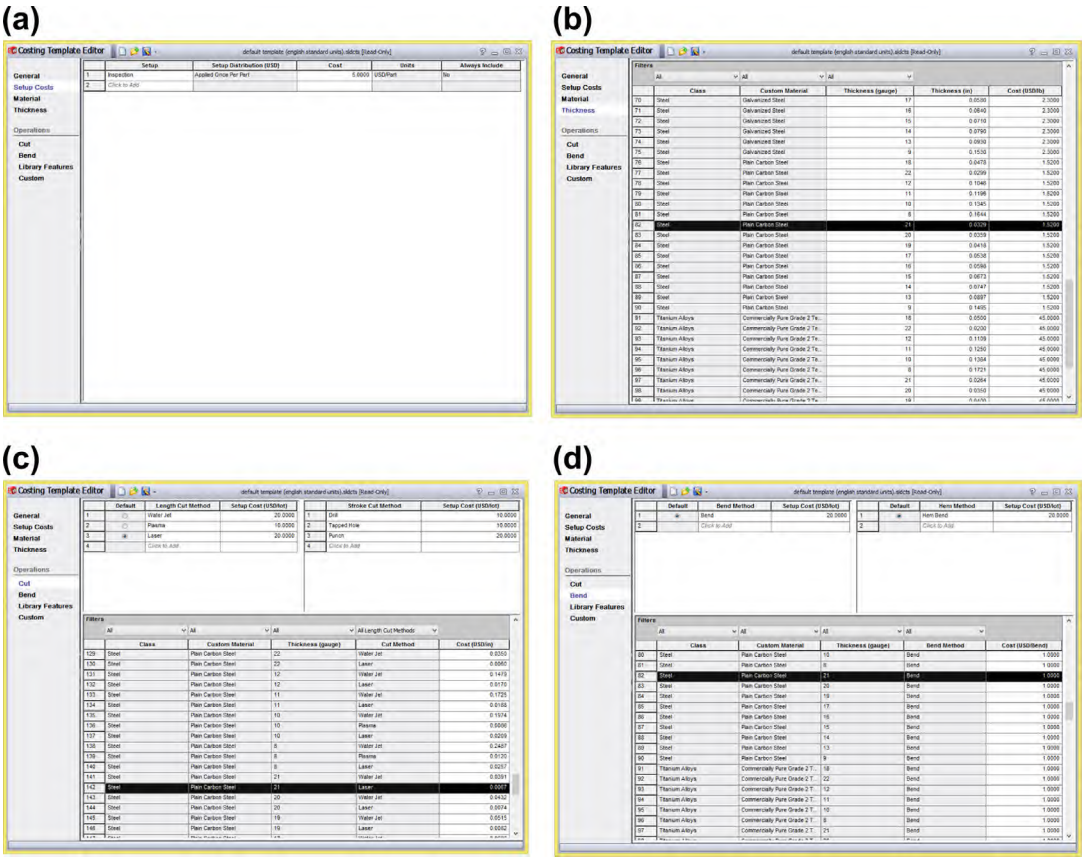


FIGURE 15.20

SolidWorks Costing templates for sheet metal: (a) Setup cost, (b) Thickness, (c) Cut, and (d) Bend.

the default templates. As a result, the estimated cost per part is $6.19; thus, material cost is $0.53 (10%) and manufacturing cost is $4.66 (90%), as shown at the bottom of the Sheet Metal Costing box in Figure 15.19(b).

Similar to the machining example, the CostingManager will appear on the left (the area of Model Tree) with manufacturing features listed, as shown in Figure 15.20(c). In the CostingManager, three major entities are listed: setup, cut paths, and bends. Under setup, two machines are listed, bend and laser, each with a cost of $0.20. Again, all cost values in the CostingManager are per part. For example, if you hover over the bend setup under Setup, the cost calculation appears as 0.2000 USD = 20.00/100 lot USD. The $20 setup cost is stored in the cost template.

How is the total cost calculated in SolidWorks Costing? There are six entities under the cut paths. Cut Path 1 cuts the center oval, Paths 2 to 5 are for the four holes, and Path 6 is cutting the outer contour of the flattened bracket. The perimeter lengths of the oval, hole, and the outer contour are 7.28 in., 1.18 in., and 25.8 in., respectively. From the Costing template of cut, using Laser to cut plain carbon

steel of thickness gauge 21 is \$0.0067/in. Therefore, the cutting costs for the center oval, hole, and the outer contour are \$0.048, \$0.00791, and \$0.180, respectively. For the bends, the cost per bend for the sheet material is \$1/bend, as stored in the cost template (Figure 15.20(d)).

Material cost can be calculated by the total weight of the sheet and the price per weight. From the Sheet Metal Costing box (Figure 15.19(b)), the area of the bounding box (rectangle) is 37.46 in.^{2} The cost of the plain carbon steel sheet is \$1.52/lb. The weight density of the plain carbon steel is 0.2818 lb./in.^{3}, and the sheet volume is 37.46 in.^{2} $\times$ 0.0329 in. $=$ 1.23 in.^{3} Thus, the material cost of the block is 1.23 in.^{3} $\times$ 0.2818 lb./in.^{3} $\times$ \$1.52/lb. $=$ \$0.528.

As with the machining costing, sheet metal costing in SolidWorks is simple and straightforward. Again, costs calculated by the Costing tool are as accurate as the data in your templates. It is critical to check the template data regularly to make sure they are accurate and up to date.

15.5.3 COST ESTIMATE FOR A BWMD USING SEER-DFM

In this case study, we use SEER-DFM, a leading commercial costing software developed by Galorath, to estimate the cost of a bicycle wind measurement device (BWMD). The bicycling accessories market is rich with creative and advanced products that appeal to casual cyclists, enthusiasts, and professionals alike. There are cycling computers that will report and log data pertaining to all manner of activity related to the bicycle (e.g., speed, distance traveled, incline, power generation, elevation, GPS location). Even the cyclist's heart rate can be captured by add-on modules and equipment. However, there is a glaring hole in the market for which there is not currently any product that measures headwind speed. As any cyclist would report, the presence and speed of a headwind can make a significant difference in the effort required to propel the bicycle forward. Therefore, a group of undergraduate students designed and prototyped a BWMD as their class project in spring 2012 (Robertson et al., 2012).

Figure 15.21(a) shows the prototype of the two major subsystems of the measurement device—the fan sensory assembly and the electronic subsystem—plus a data cable that connects these two

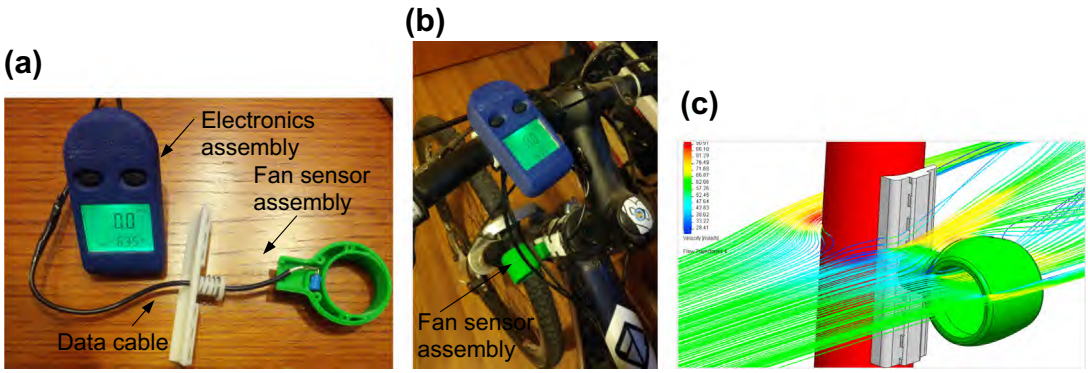


FIGURE 15.21

Prototype of the BWMD: (a) prototype assembly made by using 3D printer, (b) prototype device installed on a bicycle, and (c) CFD simulation for the fan sensor assembly.

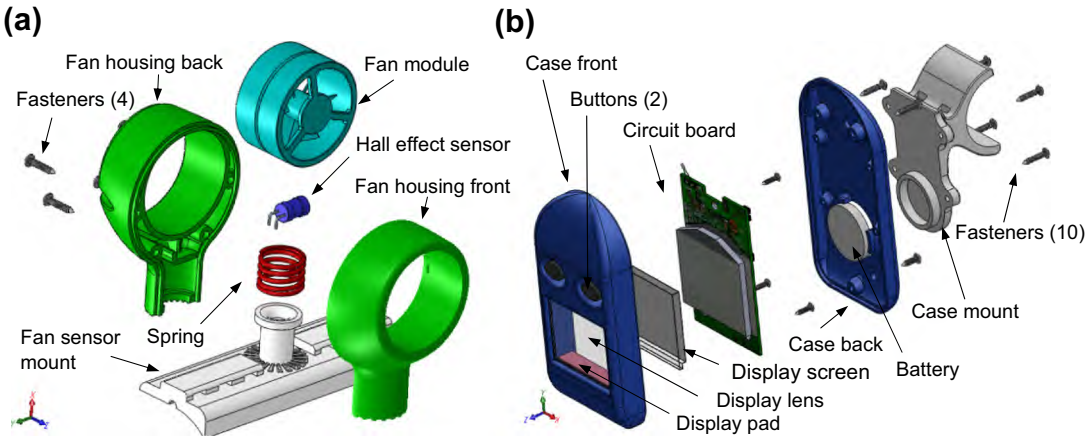


FIGURE 15.22

CAD solid models in Explode view: (a) fan sensor assembly and (b) electronics assembly.

assemblies. Figure 15.21(b) shows the two subsystems installed on a bicycle and Figure 15.21(c) illustrates a CFD simulation for the fan sensor assembly.

The CAD solid models of the two systems in explode view are shown in Figure 15.22 and a bill of materials (BOM) of the device is provided in Table 15.9. The solid models are created in SolidWorks. We assume that the factory that manufactures the device is equipped with excellent injection molding capabilities and expertise. Therefore, components are either manufactured in-house using injection molding or purchased externally.

The main window of SEER-DFM, as shown in Figure 15.23, consists of four subwindows. They are the Work Elements subwindow, which essentially contains the BOM of the product; the Inputs, which allows users to define and enter parameter values that constitute the cost of individual components as well as assembling the components; the Reports subwindow, which provides cost summary and breakdowns; and the Charts subwindow, which offers pie charts displaying the distributions of individual cost categories.

Note that all the work elements must be manually entered into the system. While creating an element, users must define the assembly level of the component in the product and choose the type of manufacturing process (e.g., injection molding, assembly, purchased part). There is no interface between SEER-DFM and CAD software.

For the BWMD, a lot size of 1000 is also assumed. In addition, to simplify the calculations, we assume that all labor costs are $50/hr. There are a total of six components manufactured using injection molding, as shown in Table 15.9. The parameters and options needed to enter for a cost estimate for the injection molding process are shown in Figure 15.24. They are entered using four input tabs: General, Processes Specific, Tooling, and Inspection/Rework.

In the General tab, we choose ABS as the material, enter 1000 for Production Quantity, and 50.00 for both Direct Hourly Labor Rate and Setup Hourly Rate, as shown in Figure 15.24(a). Note that we purposely leave the Profit (under Financial Factors) at 0.00% for the sake of calculating the true

Table 15.9 Bill of Materials and Costs Breakdown of the BWMD

Part Name	Materials	Volume (in^3)	Weight (lb)	Process	Unit Cost
1.1 Fan sensor assembly		1.58	0.0609		$40.64
1.1.1 Assembly				Assembly	$2.87
1.1.2 Fan sensor mount	ABS	0.541	0.0199	Injection molding	$12.22
1.1.3 Fan housing	ABS	0.803	0.0296	Injection molding	$12.26
1.1.4 Fan module	ABS	0.211	0.00821	Injection molding	$12.14
1.1.5 Spring	Plain carbon steel	0.0118	0.00333	Purchased	$0.25
1.1.6 Hall effect sensor		0.01025	0.000372	Purchased	$0.50
1.1.7 Fasteners (4)	Plain carbon steel	0.00136	0.000383	Purchased	$0.10
1.2 Electronic assembly		2.282	0.120	Assembly	$49.33
1.2.1 Assembly				Assembly	$5.41
1.2.2 Case front	ABS	0.580	0.0224	Injection molding	$12.22
1.2.3 Case back	ABS	0.436	0.0168	Injection molding	$12.19
1.2.4 Case mount	ABS	0.584	0.0226	Injection molding	$12.22
1.2.5 Display lens	Glass	0.102	0.00901	Purchased	$0.50
1.2.6 Display screen	Glass	0.156	0.0139	Purchased	$1.00
1.2.7 Display pad	Rubber	0.0208	0.000752	Purchased	$0.50
1.2.8 Buttons (2)	Rubber	0.0307	0.00111	Purchased	$0.20
1.2.9 Circuit board			0.0127	Purchased	$5.00
1.2.10 Battery	Chrome stainless steel	0.0605	0.0170	Purchased	$0.10
1.2.11 Fasteners (10)	Plain carbon steel	0.00136	0.000383	Purchased	$0.10
1.3 Data cable				Purchased	$0.50
Overall cost					$89.97

manufacturing cost required for these parts. In the Process Specific tab, we enter 0.3180 for Finished Weight (oz.), which in this case is the Fan Sensor Mount. Weight information must be obtained and entered for individual components. This information can be obtained using the pull-down menu: Tools > Mass Properties in SolidWorks.

In addition, number of cavities, machine capacity, and others are provided by SEER-DFM from a knowledge base implemented in the software. We stay with the default values offered by the software. In the Tooling tab, we entered 50.00 for both Tool Design Hourly Rate and Tool Fabrication Hourly Labor Rate. In the Inspection/Rework tab, we entered 5% and 2% for QA inspection and Rework, respectively. These values allow the software to calculate the costs required for QA and part rework

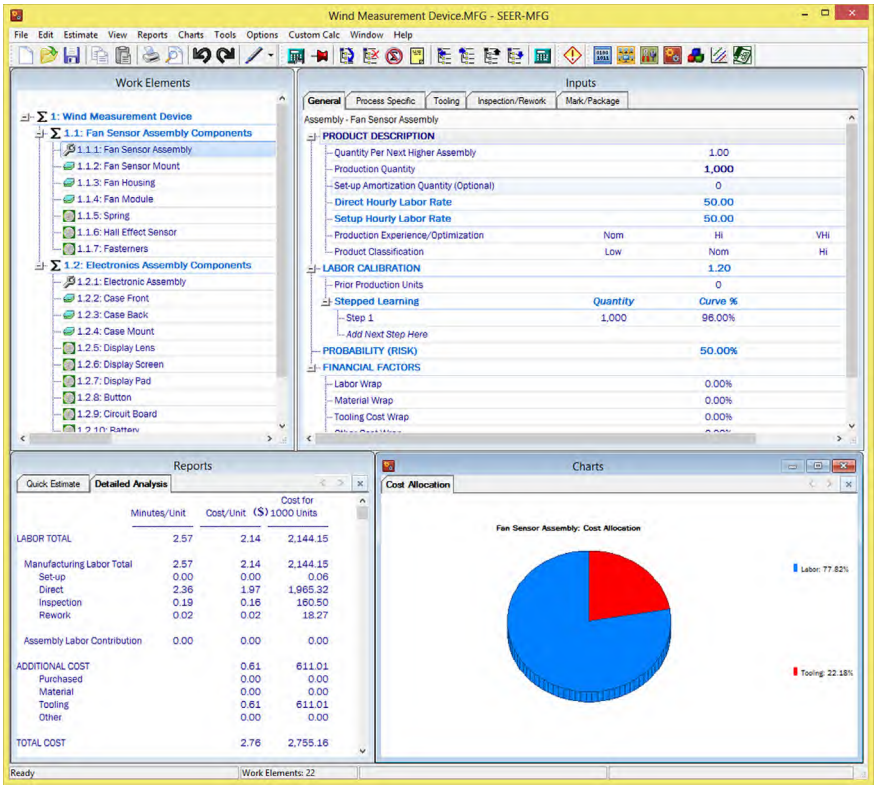


FIGURE 15.23

SEER-DFM cost-estimating software.

using the part cost as a reference. Note that we are using the same setup for all six injection molding parts.

Based on the inputs we provided, SEER-DFM estimates the unit cost of the fan sensor mount as $12.22 (Figure 15.25(a)); tooling is dominant, as shown in Figure 15.25(b). Note that similar results can be found for the other five injection molding components.

For purchased parts (e.g., spring, display lens, circuit board, fasteners), we use the General tab to enter 1000 for quantity and the Process Specific tab to enter part cost, as illustrated in Figures 15.26(a) and (b), respectively (using the spring as an example). The cost estimated, as shown in Figures 15.26(c) and (d), include only the cost required to purchase the part, as expected.

For assembly (e.g., the Fan Sensor Assembly), we use the General tab to enter 1,000 for Production Quantity and 50.00 for both Direct Hourly Labor Rate and Setup Hourly Rate, as shown in Figure 15.27(a). In the Process Specific tab, we enter all the constituent parts of the fan sensor assembly under Summary Parts List and enter 4 for fasteners, as shown in Figure 15.27(b). In the Tooling tab, we entered 50.00 for both Tool Design Hourly Rate and Tool Fabrication Hourly Labor Rate

FIGURE 15.24

Cost setup and parameter values for injection molding using the Inputs subwindow: (a) General tab, (b) Process Specific tab, (c) Tooling tab, and (d) Inspection/Rework tab.

(Figure 15.27(c)). In the Inspection/Rework tab, we enter 10% and 5% for QA inspection and Rework, respectively (Figure 15.27(d)).

Note that we use the same setup for the other subsystem—electronic assembly. The cost estimated, as shown in Figures 15.27(e) and (f), includes the labor and tooling cost required to assemble all parts for the fan sensor assembly.

The overall cost estimate is $88.97 per unit for a 1,000 lot size with labor cost rate of $50 plus tooling costs. If we increase the production quantity to 10,000 and 100,000, the unit costs become $22.16 and $16.47, respectively. This is mainly because the tooling costs were dominant for the injection molding parts; improving production quantity reduces the tooling cost per part.

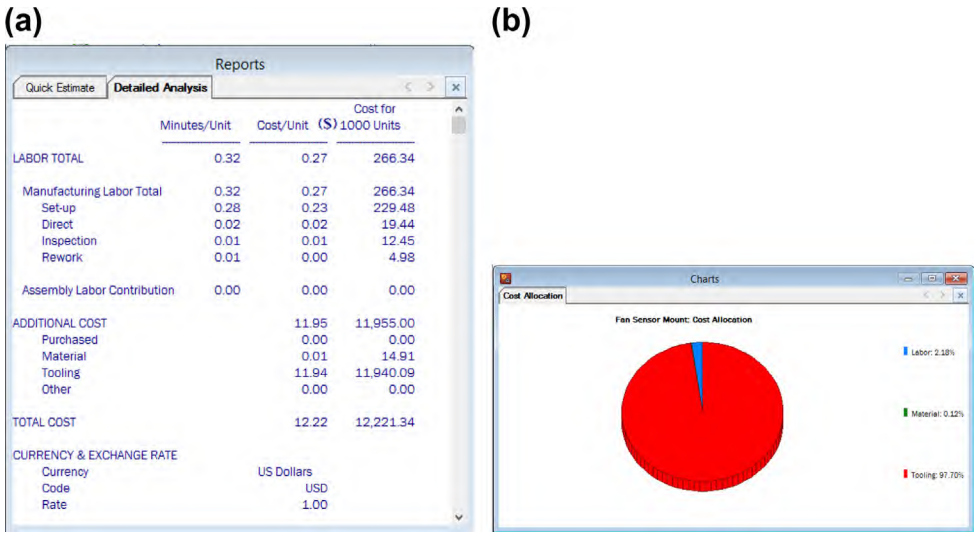


FIGURE 15.25

Cost estimate for the Fan Sensor Mount: (a) Reports of detailed analysis and (b) cost allocation chart.

As shown in this case study, SEER-DFM offers more detailed and realistic cost-estimating capabilities than that of, for example, SolidWorks. It is up to users to include as complete as possible cost data for a given product and to use as realistic as possible rate data for an accurate and realistic cost estimate.

15.6 SUMMARY

In this chapter, we introduced the important subject of the product cost estimate, which is at the heart of business practices. We covered the fundamentals of cost analysis, including cost-estimating techniques. We explained why it is critical for engineers to be able to come up with accurate estimates for product cost early in the design phase and discussed why it is difficult to do so. With the basic knowledge introduced, we narrowed our focus to discussing manufacturing cost models, including machining, injection molding, sheet metal stamping, and mechanical assembly. We also took a brief look at commercial cost-estimating software, including CAD-based, general-purpose, special-purpose, and web-based tools. Furthermore, we introduced three case studies to illustrate practical aspects of product cost estimates for engineering applications.

Product cost estimating is a substantial topic. Although we did not intend to cover a broad topic in the chapter, what is discussed is focused and practical. With what is here, readers should be able to use some of the cost models presented, case studies introduced, and/or find adequate software tool(s) for their own applications. We hope this chapter will help you develop a cost-sensitive mind-set for carrying out product design.

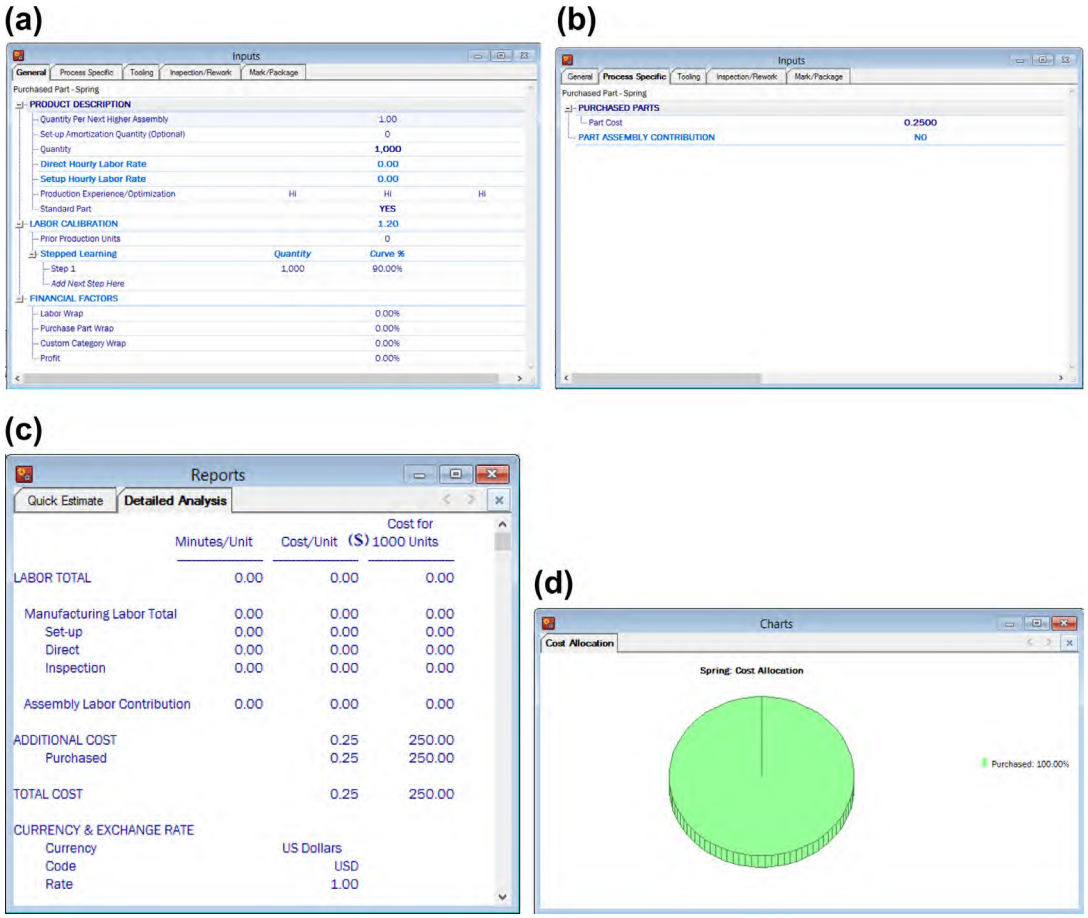


FIGURE 15.26

Cost estimate for the Spring: (a) General tab, (b) Process Specific tab, (c) Reports of detailed analysis, and (d) cost allocation chart.

APPENDIX 15A: CALCULATIONS OF MATERIAL REMOVED FOR STANDARD FEATURES

Figure 15A.1 shows a set of standard machining features that are grouped into four classes. Class 1 includes rotational features machined by using a turning process. Class 2 consists of prismatic features machined by face, slab, or end-milling operations. Group 3 includes slab features machined by end milling only. Note that end milling has two types of operations: slot end milling and peripheral end milling. Finally, Group 4 contains a revolving feature (i.e., hole) machined by a drilling operation.

FIGURE 15.27

Cost estimate for the Spring: (a) General tab, (b) Process Specific tab, (c) Tooling tab, (d) Inspection/Rework tab, (e) Reports of detailed analysis, and (f) cost allocation chart.

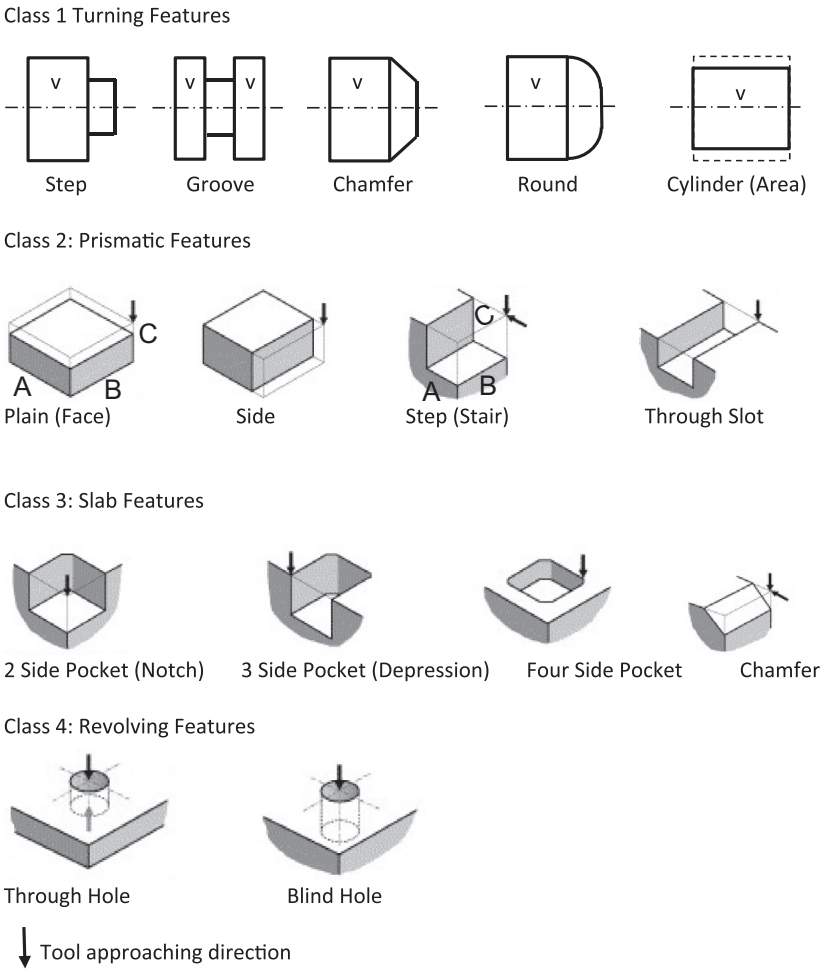


FIGURE 15A.1

A set of standard machining features.

For rough cut, the material removed, V_r, for these features can be calculated by using the equations listed in Table 15A.1. Note that for Group 1 features, the material removal rate, MRR, by turning can be calculated as

$$\text{MRR} = d\, F_r\, V \tag{15A.1}$$

where d is the depth of cut (in.), F_r is the feed/rev (in.), and V is the cutting speed (in./min.). For Group 2 features, the MRR for face milling and slot end milling is defined as

$$\text{MRR} = D\, d\, f \tag{15A.2}$$

where D is the diameter of the cutter (in.), d is the depth of cut (in.), and f is the feedrate (in./min.). For Group 3 features, the MRR for slab milling and peripheral end milling is defined

Table 15A.1 Machining Volume for Rough Cut

Feature Class	Feature Type	Volume of Rough Cut
1	Step, Groove, Cylinder	$\pi * (D1^2 - D2^2) * L/4$
	Chamfer	$\pi * (D1 * D2/4 - D2^2/12) * L$
	Round	$\pi * R1^2 * (R1 - R2) - \pi * [(R1 - R2) * R2^2 + (R1 - R2)^2 * R2 * \pi/2 + (R1 - R2)^3 * 2/3]$
2	Plain, Side, Stair. Slot	A*B*C
3	Notch, Depression, Pocket, Chamfer	A*B*C
4	Holes	$\pi * D_1^2 * H/4$

$$\text{MRR} = L\,d\,f \tag{15A.3}$$

where L is cutter length (in.). For Group 4, the MRR for slab milling and peripheral end milling is defined

$$\text{MRR} = A\,f \tag{15A.4}$$

where A is the cross-section area of a drill (in.^{2}).

The finish turning operation is carried out after the rough cut. This operation is associated with dimensional accuracy and surface finish. Therefore, operation time for finish cut is proportional to the area of the finish cut. The cut areas for standard features of Figure 15A.1 are listed in Table 15A.2. D1

Table 15A.2 Machining Area for Finish Cut

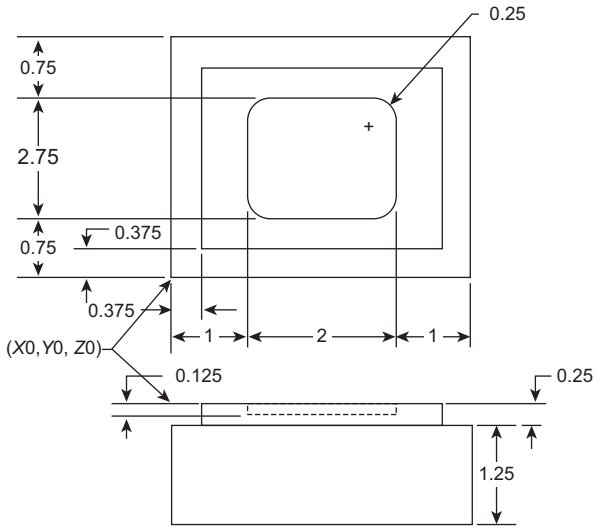
Feature Class	Feature Type	Area of Finish Cut
1	Step, Cylinder	$\pi * D2 * L + \pi * (D1^2 - D2^2)/4$
	Groove	$2 * \pi^2 + 4 * \pi * (R1 - R2)^2$
	Chamfer	$\pi * (R1 + R2) * \sqrt{L^2 + (R1 - R2)^2}$
	Round	$\pi^2 * (R1 - R2) * R2 + 2 * \pi(R1 - R2)^2$
2	Plain	A*B
	Side	B*C
	Stair	A*B + B*C
	Slot	A*B + 2B*C
3	Notch	A*B + B*C + C*A
	Depression	A*B + 2B*C + C*A
	Chamfer	$B * \sqrt{A^2 + C^2}$
4	Holes	$\pi * D_t * H$

and D2 are the outer and inner diameters of the rotational features; *L* is the length; A, B, and C are the length, width, and height of the material to be removed.

QUESTIONS AND EXERCISES

15.1. A solid block of 4 in. × 4.25 in. × 1.5 in. with a center pocket and stair at four edges shown in the following figure is to be machined from a workpiece of 4 in. × 4.25 in. × 1.5 in. The workpiece material is low-carbon steel. The pocket size is 2 in. × 2.75 in. × 0.25 in. with fillets of radius 0.25 in. at four corners. The stair feature around the edge is 0.375 in. wide and 0.25 in. deep. A ½ in. and a ¼ in. cutter of four teeth are employed for the rough and finish cuts, respectively. Calculate the cutting time for both the rough and finish cuts as well as the material cost.

15.2. If you have access to SolidWorks Costing, create a solid model like the one in Problem 1 in SolidWorks, and then use Machining Costing to estimate the manufacturing cost for the part. Conduct the same cost analysis manually and see whether the cost you obtain is identical to that of Machining Costing.



15.3. A batch of fan house front of the BWMD shown in Figure 15.22(a) is to be molded from acrylonitrile-butadiene-styrene (ABS) in a two-cavity mold, similar to the one shown in Example 15.7. Assuming that the projected area of the runner system is 10% of the projected cavity area, calculate the clamping force and select an adequate machine and machine rate. Also, estimate the mold base cost for the injection molding. Note that the CAD model of the fan house can be found at the book's companion website.

15.4. The bracket shown in Figure 15.20(a) is manufactured by stamping. In the blanking step, aluminum sheet (AL3003) is cut into one part blank per press. Calculate material cost per unit. Choose a machine and report the machine rate to cut the blank for stamping the bracket. The CAD model of the fan house can be found at the book's website.

REFERENCES

Asiedu, Y., Gu, P., 1998. Product life cycle cost analysis: state of the art review. International Journal of Production Research 36 (4), 883–908.

Bakerjian, R. (Ed.), 1992, Tool and Manufacturing Engineers Handbook Design for Manufacturability, fourth ed, vol. 6. Society of Manufacturing Engineers, Dearborn.

Black, J.T., 1991. The Design of Factory with a Future. McGraw-Hill.

Boothroyd, G., Dewhurst, P., Knight, W., 2011. Product Design for Manufacture and Assembly, third ed. CRC Press/Tylor & Francis Group.

Boothroyd, G., Reynolds, C., 1988. Approximation cost estimates for typical turned parts. Journal of Manufacturing Systems 8 (3), 185–193.

Bouaziz, Z., Ben Younes, J., Zghal, A., 2006. Cost estimation system of dies manufacturing based on the complex machining features. International Journal of Advanced Manufacturing Technology 28, 262–271.

Cavalieri, S., Maccarrone, P., Pinto, R., 2004. Parametric vs. neural network models for the estimation of production costs: A case study in the automotive industry. International Journal of Production Economics 91 (2), 165–177.

Chryssolouris, G., 2006. Manufacturing Systems: Theory and Practice, second ed. Springer-Verlag, New York.

Chryssolouris, G., Papakostas, N., Mavrikios, D., 2008. A perspective on manufacturing strategy: Produce more with less. CIRP Journal of Manufacturing Science and Technology 1, 45–52.

Clark, F., Lorenzoni, A.B., 1996. Applied Cost Engineering, third ed. CRC.

Dewhurst, P., Boothroyd, G., 1988. Early cost estimating in product design. Journal of Manufacturing Systems 7 (3), 183–191.

Dewhurst, P., Kuppurajan, K., 1987. Optimum Processing Conditions for Inject Molding, Report No. 12, Product Design for Manufacturing Series. University of Rhode Island.

Dieter, G., 1991. Engineering Design: A Materials and Processing Approach. McGraw-Hill.

Fabrycky, W.J., Blanchard, B.S., 1991. Life-Cycle Cost and Economic Analysis. Prentice-Hall.

Groover, M.P., Automation, 2001. Production Systems, and Computer-Integrated Manufacturing. Prentice-Hall.

Jung, J.Y., 2002. Manufacturing Cost estimation for machined parts based on manufacturing features. Journal of Intelligent Manufacturing 13, 227–238.

Kalpakjian, S., Schmid, S.R., 2010. Manufacturing, Engineering & Technology, sixth ed. Prentice Hall.

Lovejoy, W., Fixson, S., Jackson, S., 2010. Integrated Product Development (IPD), OMS/IOE 548, Art Des 300. University of Michigan.

Machining Data Handbook (MDH), third ed, 1980. Metcut Research Associates Inc. pp.1–233.

Meisl, C., 1988. Techniques for cost estimating in early program phases. Engineering Costs and Production Economics 14, 95–106.

Niazi, A., Dai, J.S., Balabani, S., Seneviratne, L., 2006. Product cost estimation: Technique classification and methodology review. Journal of Manufacturing Science and Engineering 128 (2), 563–576.

Ostwald, P.F., 1991. Engineering Cost Estimating, third ed. Prentice Hall.

Ostwald, P.F., McLaren, T.S., 2004. Cost Analysis and Estimating for Engineering and Management. Pearson/Prentice Hall.

Robertson, J., Walker, M., Harrison, J., Spring, 2012. Bicycle wind measurement device, Final Report. University of Oklahoma.

Roskam, J., 1985. In: Airplane Design, Volume 8. Roskam Aviation/Engineering Company.

Roy, R., Rush, C., Tuer, G., 2002. Cost estimating rational capture. In: AACE Transactions (Association for the Advancement of Cost Engineering) International Meeting. ,Oregon, Portland. June 24–26th, EST.12.1–12.10.

Rush, C., Falque, J., McRitchie, K., 2003. Real Time Cost Impact Assessment of Composite and Metallic Design Alternatives, Galorath Inc., El Segundo, CA (www.galorath.com), 3rd Annual Automotive Composites Conference, September 9–10, 2003, MSU Management Education Center, Troy, MI

Rush, C., Roy, R., 2001. Expert judgment in cost estimating: Modelling the reasoning process. Concurrent Engineering: Research and Applications (CERA) Journal 9 (4), 271–284.

Ullman, D.G., 2003. The Mechanical Design Process, third ed. McGraw-Hill.

CHAPTER

DECISIONS IN ENGINEERING DESIGN

16

CHAPTER OUTLINE

16.1 Introduction 849
16.2 Conventional Methods 850
16.2.1 Decision Matrix Method 850
16.2.2 Decision Tree Method 853
16.3 Basics of Decision Theory 857
16.3.1 Elements of a Decision 858
16.3.2 Decision-Making Models 858
16.3.3 Decision Under Risk 859
16.3.4 Decision Under Uncertainty 861
16.4 Utility Theory 864
16.4.1 Basic Assumptions 864
16.4.2 Utility Axioms 865
16.4.3 Utility Functions 866
16.4.4 Attitude Toward Risk 867
16.4.5 Construction of Utility Functions 870
16.4.6 Multiattribute Utility Functions 871
16.4.6.1 Additive MAU Functions 872
16.4.6.2 Multiplicative MAU Functions 874
16.5 Game Theory 875
16.5.1 Elements of a Game 876
16.5.2 Two-Person Matrix Games 877
16.5.3 Sequential Games 880
16.5.4 Cooperative Games 883
16.6 Design Examples 885
16.6.1 Utility Theory as a Design Tool 885
16.6.1.1 Beam Design Example 1: Unconstrained Problem 886
16.6.1.2 Beam Design Example 2: Constrained Problem 890
16.6.1.3 Summary of Utility Theory as a Design Tool 894
16.6.2 Game Theory as a Design Tool 894
16.6.2.1 The Pressure Vessel Design Example 894
16.6.2.2 Strategy Form Game 896
16.6.2.3 Sequential Game 897

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00016-8
Copyright © 2015 Elsevier Inc. All rights reserved.

16.6.2.4 Cooperative Game 900
16.6.2.5 Summary on Game Theory as Design Tool 901
16.7 Summary **901**
Questions and Exercises **902**
References **904**

Engineering design is the process by which engineers' intellect, creativity, and knowledge are translated into useful engineering products that satisfy particular functional requirements and meet engineering specifications while complying with all constraints. The traditional design approach has been one of deterministic problem-solving, typically involving efforts to meet functional requirements subject to various technical specifications and economic constraints, among others. In general, engineering design is a loosely structured, open-ended activity that includes problem definition, representation, performance evaluation, and decision making.

A number of approaches have been proposed to organize, guide, and facilitate the design process. The main objective is seeking a logical and rigorous means to aid in developing a satisfactory design, or one that is acceptable to the customer or user of the product. All approaches heavily involve decision making, which is integral to the engineering design process and is an important element in nearly all phases of design. In fact, it is fair to state that the center of all approaches is decision making.

In this chapter, we define decision making as the process of identifying and choosing alternatives from the set of possible alternatives. Alternatives are developed based on certain criteria and requirements. In the meantime, the preferences of the decision maker are incorporated to sufficiently reduce uncertainty about alternatives, which helps to achieve the desired goals and ensure a high-quality decision.

Various methods are commonly used to aid designers in decision making, such as a decision matrix (Voland, 2004), a decision tree (Eatas and Jones, 1996), quality function deployment (Akao, 2004), and so forth. These methods are generally ad hoc and incorporate relatively high levels of subjective judgment. An additional set of methods address variability, quality, and uncertainty in the design process, such as the Taguchi method (Lochner and Matar, 1990), Six Sigma (Park and Antony, 2008), Design for X (Huang, 1996), and so on. These tools are more analytical and are typically coupled to the processes used to produce products. Design theories also exist, such as Suh's axiomatic design (Suh, 1990), which are less widely used but offer more rigorous analytical bases. Finally, certain other methods are used primarily in the fields of management science and economics, such as utility and game theory, which are being explored in the current research for feasibility and applicability to support decision making in engineering design.

In the late 1990s, the National Science Foundation (NSF) initiated a series of studies to determine research priorities in engineering design by examining industry and education needs and to formulate recommendations for the NSF's Engineering Design Program (NSF, 1996). The NSF funds an online decision-based design open workshop to engage design theory researchers in a dialogue to establish a common foundation for research and educational endeavors. The NSF also sponsored Gordon Research Conferences in 1998 and 2000 on theoretical foundations to examine theories and techniques for decision making under conditions of risk, uncertainty, and conflicting human values. Such studies promoted research in design theory and led to the development of decision theories and decision-based design, including the application of utility theory and game theory to support design decision making.

For those who are interested in pursuing research topics in design theory or decision-based design, please refer to Lewis et al. (2006) for excellent discussions.

This chapter is essentially a prelude to the broad subject of design theory and methods, in which we view engineering design as a decision-making process and recognize the substantial role that decision theory can play in design. Therefore, we start by discussing basic decision methods and theory in this chapter to provide readers with an overview and broad understanding of design decision making. With such an understanding, we extend the discussion in later chapters to more practical methods and tools, such as design optimization, that are widely employed by engineers for the support of design decision making. More specifically, this chapter aims to (1) introduce basic decision theory and methods that aid readers in applying them for general decision making, (2) provide basic concepts of utility and game theories that were explored recently to aid engineering designs, and (3) use simple design examples to illustrate the concepts and methods for applying the theories to support practical engineering design problems.

16.1 INTRODUCTION

Design is a process involving constant decision making. In an engineering design context, the role of decision making can be defined in several ways. The decision process is influenced by sets of conditions or contexts; some are controllable, such as business context, and some are uncontrollable, such as market and economy conditions. The business context represents the long-term view of the company and is in general largely in the control of the company. Decisions such as capital investments, product lines and upgrades, and product marketing strategy are determined by the company. However, some aspects of business contexts, such as market share (which is influenced by competing products), are somewhat uncontrollable. Also, the state of the economy and market demands are not controlled by the company. Correctly assessing the context for making a decision is important because it dictates the level of effort and long-term impact. Decisions with long-term impacts often are irreversible after implementation; therefore, the decision maker must seriously analyze the context and impact of alternatives before arriving at a decision. A large number of short-term incremental decisions can, however, be made relatively risk-free in general.

Whether the conditions are controllable or not, there is always uncertainty involved in decision making. Narrowing the focus to product design, the uncertainty largely comes from the inputs, such as the completeness of and variation in product requirements and constraints established by the customers. Closing with the customer is an iterative process, in which reconciling the customer's needs with the developer's design capabilities require collaboration and experience with the product. Decisions made in earlier stages must be re-evaluated from time to time and adjustments need to be made in response to factors or events of high uncertainty that were not predictable or controllable throughout the design process.

In addition to the uncertainty involved in decision making, the designer's preference in choosing one alternative over another plays an important role in design, especially in dealing with multiobjective design problems, in which a designer is juggling competing objectives. In some cases, design decisions are made by design groups of a product development team, in which decisions are made to maximize their respective objectives that could be mutually competing or even conflicting.

In general in product development, decisions are made at different levels under different kinds of scenarios. At a high level, decisions are made for scenarios such as team organization, product cost,

work breakdown, and suppliers. At mid-level, a decision involves issues such as design requirements, material selection, subsystems and components, and the manufacturing process. At a low level, a designer determines design objectives, geometric shape and dimensions of the individual components, and so forth. There are many methods that aid in decision making. Some of these methods developed decades ago are more ad hoc and incorporated relatively high levels of subjective judgment, such as decision matrices, in which weighting factors that significantly impact the decision are assigned by the designer. When these methods are used, they are generally applied to support more significant project decision making at a higher level. Methods developed more recently involve rigorous theory and mathematical frameworks in decision making, such as using utility theory.

In this chapter, we intend to provide a basic introduction on both conventional methods and rigorous decision theory. We start by introducing conventional methods that are commonly employed and are easily found in design textbooks, including the decision matrix and decision tree, in Section 16.2. These methods are deterministic and rely on subjective judgment. Although we start with conventional methods, one of our focuses in this chapter is to discuss rigorous decision theory. In Section 16.3, we introduce decision theory, in which decision making is formulated and solved mathematically. In Sections 16.4 and 16.5, we discuss utility theory and game theory, respectively, which have been explored and adapted to support engineering design recently. In Section 16.6, we use simple design examples to illustrate the concept and steps in applying utility and game theories to solve simple design examples. These examples demonstrate the application of modern decision theories to support engineering design.

16.2 CONVENTIONAL METHODS

Numerous methods developed decades ago are commonly employed to aid decision making. We introduce representative and popular methods that in general support decisions at high and mid-levels, including the decision matrix and decision tree. Readers are referred elsewhere (Akao, 2004; Lochner and Matar, 1990; Park and Antony, 2008) for other popular methods, such as quality function deployment, the Taguchi method, and Six Sigma.

16.2.1 DECISION MATRIX METHOD

Decision matrix techniques are used to define attributes, weigh them, and appropriately sum the weighted attributes to give a relative ranking among design alternatives. Note that, in practice, attributes are weighted as a numeric figure based on a prescribed ranking system for individual design alternatives. In some contexts, such as design optimization, attributes are also called design objectives, which are to be maximized or minimized, or constraint functions, which must be kept within limits. In general, attributes are also referred to as design criteria or decision criteria.

A decision matrix consists of rows and columns that allow the evaluation of alternatives relative to various decision criteria. We use a material selection of airplane torque tubes shown in Figure 16.1 as an example to illustrate the method. The torque tubes are located in the front leading edge of the airplane wing, three on each side. The tubes are being redesigned to address concerns raised by the maintenance depot. The problems are twofold. First, the current magnesium tubes have poor corrosion resistance, requiring frequent repairs. Second, magnesium tubes are currently made by casting, which is extremely uneconomical when only a small quantity is required for the maintenance of the

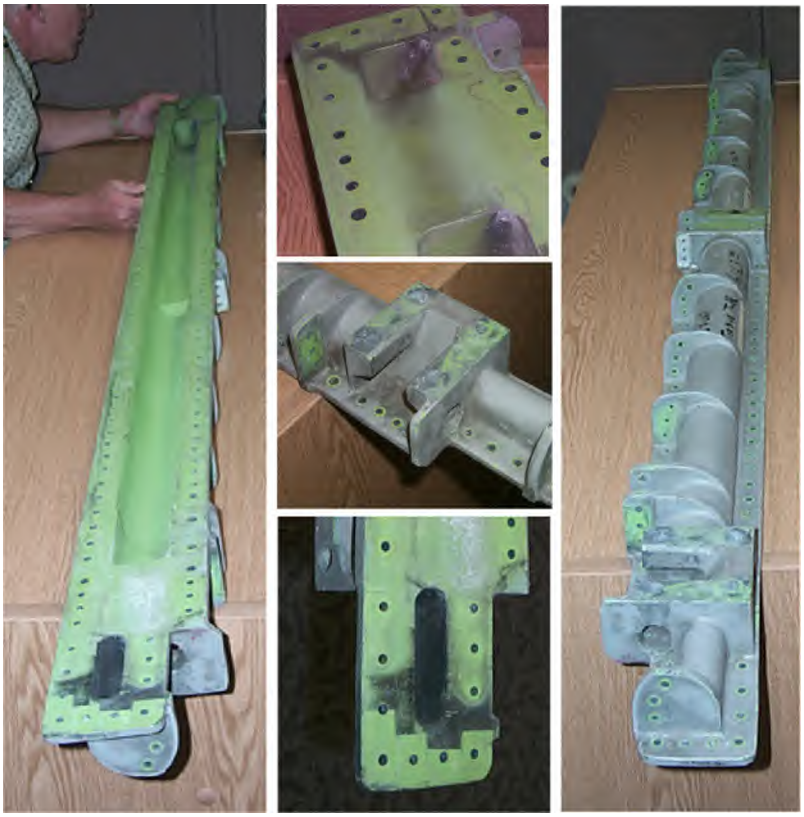


FIGURE 16.1

A sample magnesium torque tube with severe corrosion.

remaining aircraft fleet. Lead times were excessive and the cost was extremely high for acquiring the tubes. Therefore, it is desirable to manufacture the tubes using a machining process instead of casting. The goal is to choose the best possible material to replace the magnesium tubes in order to enhance product reliability and reduce manufacturing lead time, among other design criteria.

Before creating a decision matrix, a set of material options (or alternatives) and their properties relevant to the decision criteria must be identified. In this case, the general properties and cost of five common materials for aerospace applications are collected and listed in Table 16.1.

A typical decision matrix is then constructed, in which a set of design criteria, including strength, weight, machinability, corrosion resistance, and material cost, are listed in the first row of the decision matrix, as shown in Table 16.2. Note that machining cost is implicitly incorporated into the machinability criterion. We adopt a rating system of 1–5, with 1 being the worst and 5 the best, to assign a rating factor (R_f) to each material under individual criteria. Individual rating factors are assigned by referring to the properties in Table 16.1 and scaling them roughly proportionally. As shown in Table 16.2, the tallied score, the so-called decision factor (D_f), indicates that titanium Ti-6A1-4V is the best choice. However, the cost of using titanium may be too high to justify the needs.

Table 16.1 Properties and Cost of Material Options

Material	Yield Strength (ksi)	Elongation (%)	Vickers Hardness (HV)	Density (lb/in)^3	Machinability Index[a]	Galvanic Corrosion[b]	Relative Cost per Weight[c]
Magnesium AZ31B-H24	26	12	50	0.064	50	−1.6	1.8
Titanium Ti-6A1-4V	140	8	360	0.16	510	0	18
Stainless steel 430	40	20	260	0.28	165	−0.5	1
Aluminum 7075-T6	73	11	150	0.101	100	−0.8	1.4
Aluminum 2024-T4	47	19	120	0.1	110	−0.8	1.4

[a]*SAE 1117 free-machining steel has an index of 100. Higher index numbers mean that some machining operations may be more expensive to perform compared to 1117 steel.*
[b]*The Galvanic Series of Metals can be used to determine the likelihood of a galvanic reaction, and galvanic corrosion or bimetallic corrosion, between two different metals in a seawater environment. Data were taken from Atlas Steel Technical Note No. 7 "Galvanic Corrosion," with 0.3 for graphite being the best and −1.6 for magnesium being the worst.*
[c]*Data were extracted from www.roymech.co.uk/Useful_Tables/Matter/Costs.html.*

Table 16.2 Decision Matrix for Material Selection of Airplane Torque Tubes

Material	Strength	Weight	Machinability	Corrosion Resistance	Material Cost	Score
Magnesium AZ31B-H24	1	5	5	1	2	14
Titanium Ti-6A1-4V	5	3	1	5	1	15
Stainless steel 430	2	1	2	3	5	13
Aluminum 7075-T6	3	2	1.5	2	2	10.5
Aluminum 2024-T4	2	2	1.5	2	2	9.5

According to Table 16.2, the next choice is magnesium, which unfortunately is the material we want to replace to begin with.

Why does the decision matrix lead to the choice that is either too expensive or comes back to the one that we want to avoid in the first place? Certainly, one of the main issues is rating factors we assigned. For example, why is the strength factor of aluminum 7075-T6 equal to 3? Can it be 3.5? A slight adjustment in the rating factors could lead to different results. Another issue is that we assume that all design criteria have the same degree of importance. In general, they do not. In this example, corrosion and cost are considered to be the two most important criteria. In addition, weight is another important consideration for parts installed on aircrafts. Therefore, some sort of weighting scale is normally assigned to account for this variation. For this example, weighting factors 1–5 are assigned to individual design criteria, with corrosion resistance being 5, machinability and material cost that contribute largely to the overall cost being 4, and weight being 2, as shown in Table 16.3. Each weighting factor (W_f) is multiplied by the corresponding rating factor (R_f) for each material option, producing the decision factor (D_f):

$$D_f = W_f \times R_f \tag{16.1}$$

Table 16.3 Decision Matrix with Weighting Factors

Material	Strength	Weight	Machin-ability	Corrosion Resistance	Material Cost	Score
Weighting Factor	**(1)**	**(2)**	**(4)**	**(5)**	**(4)**	
Magnesium AZ31B-H24	1	5	5	1	2	44
Titanium Ti-6A1-4V	5	3	1	5	1	44
Stainless steel 430	2	1	2	3	5	47
Aluminum 7075-T6	3	2	1.5	2	2	31
Aluminum 2024-T4	2	2	1.5	2	2	30

These decision factors are then summed for each option and stored under the Score column. With the weighting factors, the best choice is revealed to be steel, although it is only marginally better than magnesium and titanium.

The decision matrix can be an important and useful tool to aid in design decision making, but the designer should be mindful that nothing takes the place of common sense and good judgment. For tools like the decision matrix to be viable, rating factors must be assigned as objectively as possible, and weighting factors must be determined to reflect the priority among design criteria. The method should help make a proper decision, rather than dictate the decision. The final decision should not be made solely on the results of the decision matrix. The value of a decision matrix is that it forces us to view the various alternatives in a careful and thoughtful manner. It is important to realize that these factors have a certain built-in uncertainty and subjectivity that may result in erroneous conclusions as to the best choice among options. A good designer will maintain a questioning attitude, always seeking further confirmation that the decision was correct as the design process evolves.

One major shortfall of the decision matrix method is that it does not take uncertainty into consideration. Uncertainty, such as the probability of the aircrafts exposed to highly corrosive operating environments, could impact the decision had the probability been known or reliably predicted in advance. Moreover, personal judgment is highly involved, especially in assigning rating factors R_f and weighting factors W_f. Different designers may come to a different design decision for the same design problem. Moreover, a designer's preference could play an important role in decision making, but it is not captured in any form.

16.2.2 DECISION TREE METHOD

The decision tree method is another way to evaluate different alternatives. This method is often used in evaluating business investment decisions, considering the outcome of possible future decisions, including the effect of uncertainties. The strength of the method is that it allows an evaluation

of the benefits of present and future profits against the investment. It is a useful technique when a decision must be made in succession into the future.

A decision tree is represented graphically using four elements:

- Branches: straight lines (—) that terminate at each end with one of three types of nodes
- Decision nodes: depicted as squares (□)
- Event (or chance) nodes: depicted as circles (○)
- Payoff nodes: depicted as price tags (⟨▭)

Every decision tree is a collection of branches connected to each other by nodes. A tree is constructed from the left end starting with a decision node. Branches emanating from a decision node represent individual options. The right end of each branch must terminate in one of the three types of nodes: decision, event (or chance), or payoff. Note that the event is also called the state of nature, which is in general out of the control of a designer.

To illustrate the method, we consider the decision tree shown in Figure 16.2, which is concerned with deciding whether the original equipment manufacturer (OEM) in the earthmoving equipment business should carry out research and development for a new product or simply improve an existing product to capture a potential market niche in the next 15–20 years. The OEM has extensive experience in developing earthmoving equipment, such as the backhoe. However, the OEM has no direct experience with the new product, which involves contour crafting that prints a house in days instead of

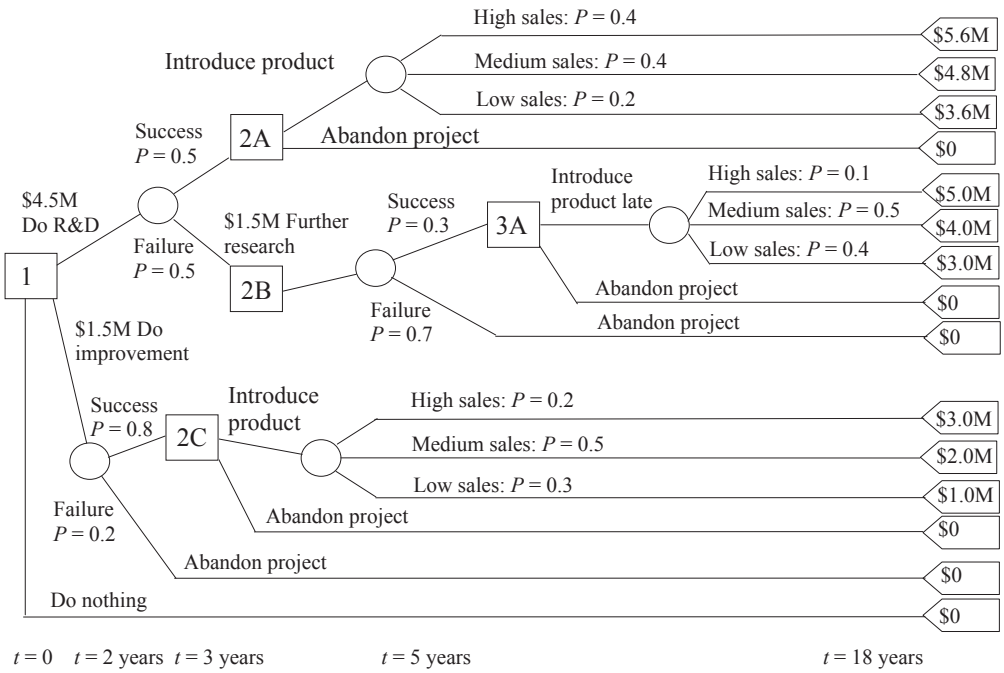


FIGURE 16.2

Decision tree for an OEM project.

months while drastically reducing material and energy consumption (see, e.g., www.contourcrafting.org). With preliminary research completed, it was found that a $4.5 million investment is required up-front to develop a new product. On the other hand, a $1.5 million investment is needed for improving an existing product. The decision for the OEM to make is do nothing, improve a current product that supports contour crafting, or develop brand-new contour crafting equipment.

As stated above, in Figure 16.2, a decision point in the decision tree is depicted by a square, and circles designate chance events that are outside the control of the decision maker. The length of line between nodes in the decision tree is not necessarily scaled with time, although the tree does depict precedence relations.

The first decision is whether to proceed with the $4.5 million project for a new product, spend $1.5 million for improving an existing product, or not do anything. If the OEM decides to proceed with the $4.5 million investment, the director of engineering department estimates that at the end of the third year, there is a 50% chance of being ready to introduce the product. If the product is introduced to the market, it is estimated to have a life of 15 years. It is also estimated by the marketing department that the probabilities of high, medium, and low sales are 40%, 40%, and 20%, respectively. The payoffs of each are estimated as $5.6 million, $4.8 million, and $3.6 million, respectively.

If the development fails to create the new product within the given time and under budget, it is estimated that an additional $1.5 million would allow the team to complete the work in an additional 2 years. The probability of successfully completing the project at the end of 5 years is 30%. Due to the delay in bringing the product to market, it is estimated by the marketing department that the probabilities of high, medium, and low sales are 10%, 50%, and 40%, respectively. It is predicted at a later time there will be more severe competition in the market. The payoffs of each are estimated as $5.0 million, $4.0 million, and $3.0 million, respectively, which are less than those of the case of successful product in 3 years due to a shorter overall product lifespan. If a product cannot be completed in 5 years, the project will have to be abandoned because there will be too much competition to introduce the product.

The other option is to improve an existing product in 2 years and introduce the product to the market 1 year earlier. This improved product will support roughly 90% of the functions envisioned in the new product. The probability of successfully completing the project at the end of 2 years is higher (80%) because the OEM has adequate knowledge and technical expertise to do so. However, because the improved product does not provide complete functions as required for contour crafting, the sales are less optimistic. The high, medium, and low sales are estimated as 20%, 50%, and 30%, respectively, with their respective payoffs to be $3.0 million, $2.0 million, and $1.0 million. If the improved product is not ready in 2 years, the management believes there is no point in continuing because the advantage of introducing the product early to the market is no longer viable.

A decision tree that incorporates the relevant information (the best information that the OEM is able to offer to the decision maker) is sketched in Figure 16.2. With the tree in place, a best possible decision can be made.

The best place to start is from the end of the branches and work backward, as illustrated in Figure 16.3. We define an expected value as in Eq. 16.2 to measure the profit or loss of individual decisions. The expected value for a decision is defined as

$$E = \sum_i P_i C_i - \sum_j I_j \tag{16.2}$$

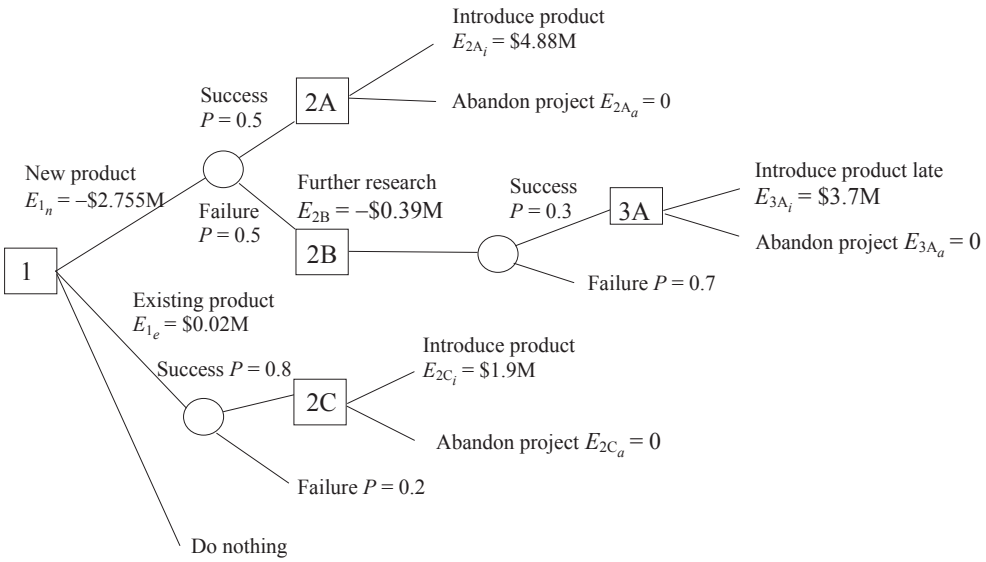


FIGURE 16.3

Solutions to the decision tree.

where P_i and C_i are respectively the probability of individual event and its payoff, and I_j is the investment. The expected values are calculated at each decision point. For example, the expected values for the decision at 2A to introduce product or abandon project in 3 years are, respectively,

$$E_{2A_i} = 0.4(\$5.6M) + 0.4(\$4.8M) + 0.2(\$3.6M) = \$4.88M$$

$$E_{2A_a} = 0$$

The expected values at decision point 3A—that is, to introduce product late or abandon the project—are, respectively,

$$E_{3A_i} = 0.1(\$5.0M) + 0.5(\$4.0M) + 0.4(\$3.0M) = \$3.7M$$

$$E_{3A_a} = 0$$

Then, at the decision point 2B, the expected value for the decision of adding \$1.5 million and introducing the product to the market at the end of the fifth year is

$$E_{2B} = [0.3(\$3.7) + 0.7(0)] - \$1.5M = -\$0.39M$$

Thus, carrying out the analysis for the delayed project back to 2B shows that to continue the project beyond that point results in a negative expected value. The proper decision, therefore, is to abandon the project if it is not successful in the first 3 years to cut the losses.

Now, we calculate the expected values for the option of improving an existing product. At the decision point 2C, the expected values of introducing the product or abandoning the project are, respectively,

$$E_{2C_i} = 0.2(\$3.0M) + 0.5(\$2.0M) + 0.3(\$1.0M) = \$1.9M$$

$$E_{2C_a} = 0$$

Finally, the expected value for the on-time project in introducing a new product at decision point 1 is

$$E_{1_n} = 0.5(\$4.88\text{M}) + 0.5(-\$0.39\text{M}) - \$5\text{M} = -\$2.755\text{M}$$

in which the large negative value is due to either the expected payoffs being too modest or the cost for the new product development being too great to be warranted by the payoff.

The expected value for a successful project in improving an existing product at decision point 1 is

$$E_{1_e} = 0.8(\$1.9\text{M}) - \$1.5\text{M} = \$0.02\text{M}$$

indicating a small margin of profit can be expected. Therefore, based on the estimates of payoffs, probabilities, and costs, the OEM should proceed with the option of improving an existing product, instead of developing a new product.

As illustrated in this example, the decision tree is a useful and effective tool in support of business-type decision making. However, the expected values that the final decision is made upon are dependent on the estimates of payoffs, probabilities, and costs. These estimates are highly uncertain to say the least, although the engineering and marketing departments might have completed adequate research in coming up with these estimates. For example, if a competitor introduces a similar product to the market, the probabilities of sales and payoffs are mostly negatively impacted. Furthermore, the impact of later decisions (and the information generated by them) on earlier decisions is important in the effective use of decision trees. It is important for the decision tree to be regarded as a dynamic device, one in which new (and better) information is integrated as it becomes available.

Although we bring in probability into the decision tree and conduct so-called decision making under risk, the probabilities are assumed to be known in advance. This is not the case for many decision-making scenarios in practice. In the following sections, we introduce decision theory, which provides a rational and more rigorous framework for the support of decision making. We also introduce theory and methods that help decision making under numerous situations that are more general and practical.

16.3 BASICS OF DECISION THEORY

Decision theory provides a rational framework for choosing between alternative courses of action when the consequences resulting from this choice are imperfectly known (North, 1968). Decision theory has been applied more to business management situations than to engineering design decisions. The purpose of this section is to acquaint the reader with the basic concepts of decision theory, including elements of a decision, decision criteria, decision under uncertainty, and attitude toward risk. Later, in Sections 16.4 and 16.5, we discuss two major theories extended from the basics of decision theory—utility theory and game theory. These theories were explored for support of engineering design in recent years. We also present examples to illustrate the concepts and steps of applying these theories for simple and yet practical engineering design problems in Section 16.6.

In this section, we start with a simple example that involves a decision for buying a new or used car. The major criteria for making such a decision are cost and reliability. The car buyer wants to buy a reliable car with lesser cost that lasts for 10 years without major problems, such as engine overhaul or transmission repairs. To simplify our discussion, we assume the car buyer is interested only in one specific car model of new or used. Buying a new car will cost more at the beginning, but the probability of having major problems in 10 years is less. On the other hand, buying a used car may be cheaper up-front, but the probability of encountering major problems in the 10-year period is higher, which may

increase the overall cost at the end. We use this example to illustrate terminologies and basic concepts to facilitate later discussions.

16.3.1 ELEMENTS OF A DECISION

There are three basic elements to any decision: courses of action, the state of nature, and the payoffs. Courses of action (a_i), also called acts or alternatives, are the choices available, which are under the control of the decision maker—for example, the choice of buying a new or used car, as discussed. States of nature (ϕ_j), also called events, is an exhaustive list of possible future events. The decision maker has no direct control over the occurrence of a particular event. The car buyer does not know whether the car he or she buys will encounter major problems in the next 10 years. Payoff (v), also called outcome, quantifies effectiveness associated with a specified combination of a course of action and state of nature. Payoff is often arranged in a summary table.

Returning to the car-buying example, let us say that the buyer did some research and was referred to online consumer reports, which provided basic information about the maintenance data and major repair costs of the specific car model in which the buyer is interested. Based on the information the buyer came up with, the estimates of the overall cost of new and used cars for a 10-year period, including purchase price, expenses (gas), regular maintenance (oil change, brake pads, air filters, and so on), and repair cost (including major repairs), are listed in Table 16.4. This is an example of a payoff table. In general, a payoff table can be generalized and formulated such as that in Table 16.5, where a_i is the ith alternate course of action, ϕ_j is the jth state of nature, and v is the payoff or value associated with a specified combination of a course of action and state of nature.

In many cases, we can make better decisions if we establish probabilities for the states of nature. These probabilities may be based on historical data or subjective estimates, which are less desirable. They are often referred to as states of knowledge, which describe the degree of certainty that can be associated with the respective states of nature. The state of knowledge affects significantly how the decision is made, which is discussed in the following subsections.

16.3.2 DECISION-MAKING MODELS

Decision-making models usually are classified with respect to the state of knowledge. Depending on the state of knowledge, decisions are made under certainty, under risk, under uncertainty, and under conflict.

Decision under certainty implies that each action results in a known outcome that will occur with a probability of 100%. Action refers to a choice of alternatives. For example, a designer may choose one of the five materials listed in Table 16.1. The outcome of an action is assumed to be known. The design

Table 16.4 Cost of Different Options and Events (Payoff Table)

States of Nature (Events)	Courses of Action	
	New Car (N)	Used Car (U)
Without major problems (W)	\$25,000	\$15,000
With major problems (F)	\$35,000	\$30,000

Table 16.5 Generic Payoff Table

States of Nature (Events)	Courses of Action			
	a_1	a_2	...	a_m
ϕ_1	$v(a_1, \phi_1)$	$v(a_2, \phi_1)$	...	$v(a_m, \phi_1)$
ϕ_2	$v(a_1, \phi_2)$	$v(a_2, \phi_2)$	...	$v(a_m, \phi_2)$
...	...	...	...	...
ϕ_n	$v(a_1, \phi_n)$	$v(a_2, \phi_n)$	...	$v(a_m, \phi_n)$

matrix method discussed in Section 16.2.1, which assumes definite rating factors and weighting factors, is a typical example of decision making under certainty.

Decision under risk assumes that each action results in known outcomes that will occur with a known probability, which is less than 100%. The decision tree discussed in Section 16.2.2 supports decision making with assigned probabilities for events that branch out at chance nodes. In the OEM example discussed in Section 16.2.2, the director of the engineering department estimated that, at the end of the third year, there was a 50% chance of the new product being ready to enter the market, if the OEM decided to proceed with the $4.5 million investment. Although such decision making is risky, the outcomes and probabilities are assumed to be known.

Decision under uncertainty implies that the probabilities that affect the outcomes usually are not known with much confidence. This presents a more realistic situation. Therefore, the decision maker should concentrate on achieving the best estimate of those probabilities. A formal theory of decision making that takes uncertainty into consideration is the utility theory, which will be discussed in Section 16.4.

Decision under conflict refers to a situation where decisions are made by more than one person or teams simultaneously or sequentially, and each decision maker or team is trying to maximize one's own objectives. This type of decision is supported by game theory, which will be briefly discussed in Section 16.5.

The feasibility of applying utility theory and game theory to support engineering design was explored and documented in details (e.g., Lewis et al., 2006). We use simple design examples to illustrate the concept in Section 16.6.

Before we move on to the utility theory and game theory, we introduce a few more basic concepts on the decision theory. We focus on two decision models: decision under risk and decision under uncertainty. We use the simple car-buying example to illustrate the decision models. In addition, we introduce utility functions that characterize the decision maker's attitude toward risk, which is essential to understanding the utility theory to be introduced later.

16.3.3 DECISION UNDER RISK

In this section, we revisit the decision under risk, in which the probabilities that affect the outcomes usually are assumed to be known. We introduce more rigorous treatment to the decision-making model, in which we revisit the car-buying example for illustration.

To facilitate our discussion, we assign N and U to the options of new and used car, respectively; and W and F to the events of car works well without major repairs and car fails due to major problems,

Table 16.6 Cost of Different Options and Events (Payoff Table)

States of Nature (Events)	Courses of Action					
	New Car (N)			Used Car (U)		
	Probability	Payoff	Expected Value	Probability	Payoff	Expected Value
Without major problems (W)	$P(W\|N) = 0.8$	\$25,000	\$20,000	$P(W\|U) = 0.5$	\$15,000	\$7500
With major problems (F)	$P(F\|N) = 0.2$	\$35,000	\$7000	$P(F\|U) = 0.5$	\$30,000	\$15,000
		\$27,000			\$22,500	

respectively. The buyer also consulted with the dealership, and the historical data suggest that there is an 80% chance that a new car does not encounter major problems in 10 years. The historical data also suggest that the probability is 50% for a used car. At the time, this was the best possible estimate that the buyer was able to attain. Next, we construct a decision tree to aid the buyer in making a decision. The payoff table of Table 16.4 is expanded by incorporating the probabilities and expected values. In Table 16.6, the notation $P(W|N)$ stands for the probability of new car (N) that works (W) well in 10 years. Similarly, $P(F|U)$ stands for the probability of used car (U) that fails (F) in 10 years (i.e., encountering major problems), and so on.

Assuming these historical data are reliable, the payoff table and the probability estimates can be combined to arrive at the expected payoff of individual decisions. The expected payoff is also called the expected monetary value (EMV) in decision theory. The calculations are summarized as follows:

$$E(a_i) = \sum_j P(\phi_j) v(a_i, \phi_j) \tag{16.3}$$

where $E(a_i)$ is the EMV of event a_i.

Hence, for the event of buying a new car, the expected payoff is

$$\begin{aligned} E(N) = \sum_j P(\phi_j) v(N, \phi_j) &= P(W|N)v(N, W) + P(F|N)v(N, F) \\ &= 0.8(\$25{,}000) + 0.2(\$35{,}000) = \$27{,}000 \end{aligned}$$

Similarly, for the event of buying a used car, the expected payoff is $E(U) = \$22{,}500$, as shown in Table 16.6. Therefore, based on the expected payoff, buying a used car presents a better option.

The car-buying example can be represented in a decision tree similar to that of Section 16.2.2. The decision tree for the car-buying example is shown in Figure 16.4.

As discussed in Section 16.2.2, the general approach to solving a decision tree is to move backward through the tree (from right to left) until we reach the originating decision node. We select a payoff node and move left to trace the branch to encounter the next node. If the next node is a chance node, we calculate the expected value (E) of all nodes connected immediately to the right of the encountered node by using Eq. 16.3. In this example, we calculate the expected values for the options of buying a new car $E(N)$ and buying a used car $E(U)$, respectively.

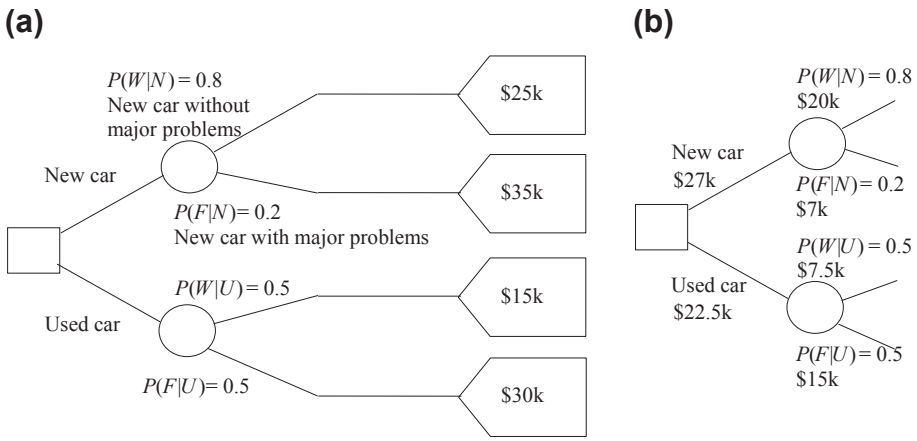


FIGURE 16.4

Decision tree for a car buyer: (a) starting tree and (b) solution tree.

We enter these values to their respective chance nodes, and then move left to encounter a decision node—in this case, the originating decision node, as shown in Figure 16.4(b). At the decision node, we select the branch that leads to the best value. In this case, \$22,500 is the best value, representing a lesser overall cost. At this point, we reach a decision of buying a used car.

One of the difficulties in using the decision tree method is coming up with the probabilities of the uncertain event occurring. In the car-buying example, how certain is the event that the probability of a used car without major problem is $P(W|U) = 50\%$? In this case, we do not have to agonize too much over the accuracy of the estimate because we can easily test to see if the decision to buy a used car is sensitive to the probability estimated. We first let the probability be $P(W|U) = x$. Then the probability of a used car having major problems is $P(F|U) = 1 - x$. The expected value of buying a used car is

$$E(U) = x(\$15\text{k}) + (1 - x)(\$30\text{k}) = \$30\text{k} - \$15\text{k}\, x$$

If we equate the two expected values of used and new car $E(U) = E(N) = \$27\text{k}$, we have $x = 0.2$. In other words, as long as the probability of a used car without major problems is greater than 20%, buying a used car is still a good choice.

16.3.4 DECISION UNDER UNCERTAINTY

In this section, we discuss decisions under uncertainty, in which the probabilities that affect the outcomes usually are not known with much confidence. This is the most general and practical decision model. The essence of this decision model is improving the confidence level of the knowledge of the states.

We introduce the Bayesian approach to decision making, in which the improved estimation of probabilities is achieved by using Bayes' theorem, which is a means of revising prior estimates of probabilities on the basis of new information. Certainly, it is logical, when faced with making a decision with uncertainty present, to try to remove the elements of uncertainty (or minimize their impact)

by gathering more information about the nature of the events. The more knowledge we possess, the less the uncertainty. However, there usually are real limits of cost and time to achieve complete knowledge. The Bayesian approach is a very satisfactory compromise by which we combine additional knowledge with our initial estimation (prior probabilities) to form revised probabilities (posterior probabilities) that, in turn, provide us with a revised basis on which to make our decision.

Bayes' theorem was discussed in Chapter 10 Reliability Analysis of Part II Product Performance Evaluation. Readers are encouraged to review that chapter to gain a basic understanding on the concepts of probability. Here, we simply state the theorem (Eq. 10.22) below:

$$P(E_i|A) = \frac{P(A|E_i)P(E_i)}{P(A)} = \frac{P(A|E_i)P(E_i)}{\sum_{i=1}^{m} P(A|E_i)P(E_i)} \tag{16.4}$$

where $P(E_i)$ is the probability of event E_i, $P(E_i|A)$ is the conditional probability of event E_i when event A has occurred, and $P(A|E_i)$ is the conditional probability of event A given that event E_i has occurred. We assume that events E_1, E_2, …, E_m are mutually exclusive.

Now let us revisit the car-buying example. Instead of the buyer conducting research on his or her own to come up with the estimates on the probabilities, the dealership owns data accumulated for many years through offering services to the specific car model the buyer is interested. We assume that the buyer is willing to pay \$1000 to purchase the data from the dealership, and the dealership agrees to sell the data to the buyer for \$1000. The dealership data show that there are approximately 150,000 cars of the model on the road as of today. Among these cars, 70% of them have not encountered major problems in the past 10 years (i.e., $P(W) = 0.7$), and the remaining 30% either went through engine overhauls or transmission repairs in the same time period ($P(F) = 0.3$). Among the cars without major problems, 95% were bought as brand new ($P(N|W) = 0.95$) and 5% were used ($P(U|W) = 0.05$). Among the cars with major problems, 15% of them were bought as brand new car ($P(N|F) = 0.15$) and 85% were used ($P(U|F) = 0.85$). These data are illustrated in Figure 16.5 as a probability tree.

As illustrated in the probability tree, the probabilities of new car with and without major problems can be calculated using Eq. 16.4 as, respectively,

$$P(F|N) = \frac{P(N|F)P(F)}{P(N|W)P(W) + P(N|F)P(F)} = \frac{0.045}{0.665 + 0.045} = 0.0634$$

$$P(W|N) = \frac{P(N|W)P(W)}{P(N|W)P(W) + P(N|F)P(F)} = \frac{0.665}{0.665 + 0.045} = 0.937$$

and the probabilities of used car with and without major problem can be calculated using Eq. 16.4 as, respectively,

$$P(F|U) = \frac{P(U|F)P(F)}{P(U|W)P(W) + P(U|F)P(F)} = \frac{0.225}{0.225 + 0.035} = 0.865$$

$$P(W|U) = \frac{P(U|W)P(W)}{P(U|W)P(W) + P(U|F)P(F)} = \frac{0.035}{0.225 + 0.035} = 0.135$$

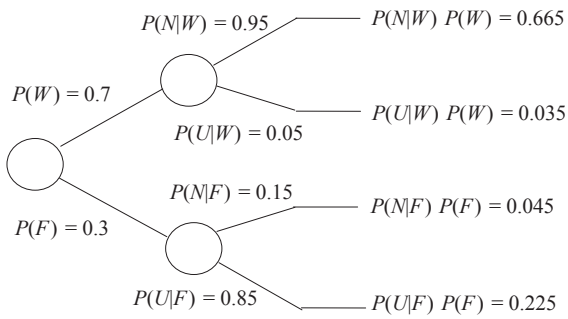


FIGURE 16.5

Probability tree for car failure data.

With the improved probability data, we revisit the decision tree shown in Figure 16.4. Following the probability data shown in Figure 16.5, the expected values of the chance nodes on top (buying new car) and bottom (buying used car), as shown in Figure 16.6, become

$$E(N) = 0.937(\$26\text{k}) + 0.0634(\$36\text{k}) = \$26.64\text{k}$$

and

$$E(U) = 0.135(\$16\text{k}) + 0.865(\$31\text{k}) = \$28.98\text{k}$$

in which the \$1000 expense of data purchasing has been added to the payoffs. The results show that buying a new car is a better and economical option. Thus, by using posterior probability, we increased our knowledge of the probability that leads to a decision of buying a new car in contrast to the earlier decision.

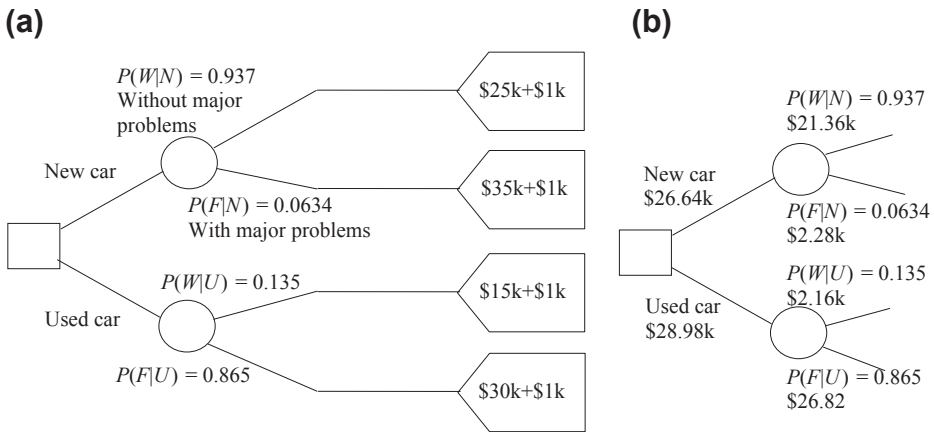


FIGURE 16.6

Decision tree for a car buyer with posterior probability: (a) starting tree and (b) solution tree.

Using posterior probability, we increased our knowledge of the probability that could help us arrive at a better decision. However, in our discussion so far, we assume the payoff of individual events is precisely known. The question is: Does driving a new car without encountering a major problem for 10 years really cost $25,000? Similarly, does a failed used car really cost $30,000? In reality, we do not have precise knowledge in these outcomes. In addition, a person might prefer buying a new car to getting a used car because he or she feels safer driving a new car. After all, the decision is being made by a human being. Personal preferences that could play an important role in decision making have not been taken into consideration. We address these issues next.

16.4 UTILITY THEORY

In Sections 16.3.3 and 16.3.4, we used an expected value rule to support decision making under risk and uncertainty. In many situations, it is highly desired that a decision maker's preference is incorporated. Utility theory offers a rigorous mathematical framework, within which we are able to examine the preferences of individuals and incorporate them into decision making. In this section, we discuss utility theory from a design perspective. We discuss basics of the theory, including assumptions that lead to axioms of the theory, the utility functions that capture an individual's attitude toward risk, and the construction of utility functions for single and multiple attributes. In the context of design, the attributes are design criteria or design objectives to attain.

16.4.1 BASIC ASSUMPTIONS

There are four assumptions that lead to basic axioms of the utility theory. The first and perhaps the biggest assumption to be made is that any two possible outcomes resulting from a decision can be compared. Given any two possible outcomes, the decision maker can say which one he or she prefers. In some cases, the decision maker can say that they are equally desirable or undesirable. A reasonable extension of the existence of one's preference among outcomes is that the preference is transitive; that is, if one prefers A to B and B to C, then it follows that one prefers A to C.

The second assumption, originated by von Neumann and Morgenstern (1947), forms the core of modern utility theory. This assumption states that one can assign preferences in the same manner to lotteries involving prizes as one can to the prizes themselves. The lottery gives the probability of one getting prize A is p, and the probability that one gets prize B is $1 - p$. Such a lottery is denoted as $(p, \text{A}; 1 - p, \text{B})$, as represented in Figure 16.7, in which p is between 0 and 1. In utility theory, uncertainty is modeled through lotteries.

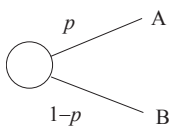


FIGURE 16.7

Lottery diagram.

Now suppose one is asked to state his or her preferences for prize A, prize B, and a lottery of the above type (p, A; $1-p$, B). Let us assume one prefers prize A to prize B. Then, based on von Neumann and Morgenstern, one would prefer prize A to the lottery (p, A; $1-p$, B) because there is a probability $1-p$ that one would be getting the inferior prize B of the lottery. One would also prefer the lottery (p, A; $1-p$, B) to prize B for all probabilities p between 0 and 1. In other words, one would rather have the preferred prize A than the lottery, and one would rather have the lottery than the inferior prize B. Furthermore, it seems logical that, given a choice between two lotteries involving prizes A and B, one would choose the lottery with the higher probability of getting the preferred prize A. That is, one prefers lottery (p, A; $1-p$, B) to (p', A; $1-p'$, B) if and only if p is greater than p'.

The third assumption is that one's preferences are not affected by the way in which the uncertainty is resolved, bit by bit, or all at once. To illustrate the assumption, let us consider a compound lottery—a lottery in which at least one of the prizes is not an outcome but another lottery among outcomes. For example, consider the lottery (p, A; $1-p$, (p', B; $1-p'$, C)), as depicted in Figure 16.8(a). According to the third assumption, one can decompose a compound lottery by multiplying the probability of the lottery prize in the first lottery by the probabilities of individual prizes in the second lottery. One should be indifferent between (p, A; $1-p$, (p', B; $1-p'$, C)) and (p, A; $(1-p)p'$, B; $(1-p)\,(1-p')$, C), as depicted in Figure 16.8(b).

The fourth assumption is continuity. Consider three prizes: A, B, and C. One prefers A to C, and C to B. We shall assert that there must exist some probability p so that one is indifferent to receiving prize C or the lottery (p, A; $1-p$, B) between A and B. C is called the certain equivalent of the lottery (p, A; $1-p$, B). In other words, if prize A is preferred to prize C and C is preferred to prize B, for some p between 0 and 1, there exists a lottery (p, A; $1-p$, B) such that one is indifferent between this lottery and prize C.

16.4.2 UTILITY AXIOMS

We now summarize the assumptions we have made into the following axioms. We have prizes or outcomes A, B, and C from a decision. We use the following notations:

$\succ$ means "is preferred to," for example, A $\succ$ B means A is preferred to B.
$\sim$ means "is indifferent to," for example, A $\sim$ B means the decision maker is indifferent between A and B.

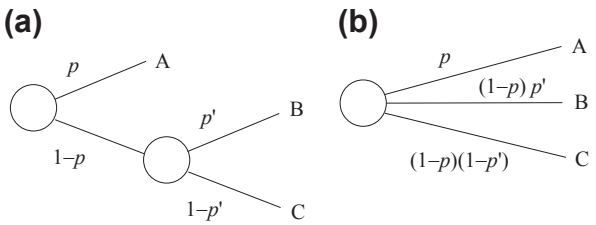


FIGURE 16.8

Lottery diagrams: (a) compound lottery and (b) equivalent simple lottery.

There are six axioms that serve the basis of the utility theory. They are orderability, transitivity, continuity, substitutability, monotonicity, and decomposability, which are defined as follows:

Orderability: Given any two prizes or outcomes, a rational person prefers one of them, else the two are equally preferable. Mathematically, the axiom is written as

$$(\mathrm{A} \succ \mathrm{B}) \vee (\mathrm{B} \succ \mathrm{A}) \vee (\mathrm{A} \sim \mathrm{B}) \tag{16.5}$$

in which "$\vee$" means "or." Eq. 16.5 reads A is preferred to B, or B is preferred to A, or A and B are indifferent.

Transitivity: Preferences can be established between prizes and lotteries in an unambiguous fashion. The preferences are transitive; that is, given any three prizes or outcomes A, B, and C, if one prefers A to B and prefers B to C, one must prefer A to C. Mathematically, we have

$$(\mathrm{A} \succ \mathrm{B}) \wedge (\mathrm{B} \succ \mathrm{C}) \Rightarrow (\mathrm{A} \succ \mathrm{C}) \tag{16.6}$$

in which "$\wedge$" means "and," and "$\Rightarrow$" means "implies."

Continuity: If $\mathrm{A} \succ \mathrm{C} \succ \mathrm{B}$, there exists a real number p with $0 < p < 1$ such that $\mathrm{C} \sim (p, \mathrm{A}; 1-p, \mathrm{B})$. That is, it makes no difference to the decision maker whether C or the lottery $(p, \mathrm{A}; 1-p, \mathrm{B})$ is offered to him or her as a prize. Mathematically, we have

$$(\mathrm{A} \succ \mathrm{C} \succ \mathrm{B}) \Rightarrow \exists \ p \in (0,1) | (p, \mathrm{A}; \ 1-p, \mathrm{B}) \sim \mathrm{C} \tag{16.7}$$

in which "$\exists$" means "there exists" and "|" means "such that."

Substitutability: If one is indifferent between two lotteries, A and B, then there is a more complex lottery in which A can be substituted with B. Mathematically, we have

$$(\mathrm{A} \sim \mathrm{B}) \Rightarrow (p, \mathrm{A}; \ (1-p), \ \mathrm{C}) \sim (p, \ \mathrm{B}; \ (1-p), \ \mathrm{C}) \tag{16.8}$$

Monotonicity: If one prefers A to B, then one must prefer the lottery in which A occurs with a higher probability; in other words, if $\mathrm{A} \succ \mathrm{B}$, then $(p, \mathrm{A}; 1-p, \mathrm{B}) \succ (p', \mathrm{A}; 1-p', \mathrm{B})$ if and only if $p > p'$. Mathematically, we have

$$(\mathrm{A} \succ \mathrm{B}) \Rightarrow (p > p' \Leftrightarrow (p, \mathrm{A}; \ 1-p, \ \mathrm{B}) \succ (p', \ \mathrm{A}; \ 1-p', \ \mathrm{B})) \tag{16.9}$$

in which "$\Leftrightarrow$" means "if and only if."

Decomposability: Compound lotteries can be reduced to simpler lotteries using the laws of probability; that is,

$$(p, \mathrm{A}; \ 1-p, (p', \mathrm{B}; \ 1-p', \ \mathrm{C})) \sim (p, \ \mathrm{A}; \ (1-p)p', \ \mathrm{B}; \ (1-p)(1-p'), \ \mathrm{C}) \tag{16.10}$$

16.4.3 UTILITY FUNCTIONS

If a decision maker obeys the axioms of the utility theory, there is a concise mathematical representation possible for preferences: a utility function $u(\cdot)$ that assigns a number to each lottery or prize. The utility function has the following properties:

$$u(\mathrm{A}) > u(\mathrm{B}) \Leftrightarrow \mathrm{A} \succ \mathrm{B}; \ \text{ and } \ u(\mathrm{A}) = u(\mathrm{B}) \Leftrightarrow \mathrm{A} \sim \mathrm{B} \tag{16.11}$$

and

$$C \sim (p, A; \ 1-p, \ \mathrm{B}) \Rightarrow u(\mathrm{C}) = p\, u(\mathrm{A}) + (1-p)\, u(\mathrm{B}) \tag{16.12}$$

which implies that the utility of a lottery is the mathematical expectation of the utility of the prizes. It is this "expected value" property that makes a utility function useful because it allows complicated lotteries to be evaluated quite easily.

It is important to realize that all the utility function does is offer a means of consistently describing the decision maker's preferences through a scale of real numbers, provided that these preferences are consistent with the first four previously mentioned assumptions. The utility function is no more than a means to logical deduction based on given preferences. The preferences come first and the utility function is only a convenient means of describing them.

16.4.4 ATTITUDE TOWARD RISK

In Section 16.4.3, we assume a decision maker makes the decisions in the presence of uncertainty by maximizing its expected utility. In many situations, this rule does not adequately model the choices most people actually make. One famous example is the St. Petersburg lottery or St. Petersburg paradox, which is related to probability and decision theory in economics.

The St. Petersburg paradox, first published by Daniel Bernoulli in 1738 (Sommer, 1954), is a situation where a naive decision criterion that takes only the expected value into account predicts a course of action that presumably no actual person would be willing to take. The paradox is illustrated as follows. A casino offers a game of chance for a single player in which a fair coin is tossed at each stage. The pot starts at \$1 and is doubled every time a head appears. The first time a tail appears, the game ends and the player wins whatever is in the pot. Thus, the player wins \$1 if a tail appears on the first toss, \$2 if a head appears on the first toss and a tail on the second, \$4 if a head appears on the first two tosses and a tail on the third, \$8 if a head appears on the first three tosses and a tail on the fourth, and so on. In short, the player wins $\$2^n$, where n heads are tossed before the first tail appears. What would be a fair price to pay the casino for entering the game?

To answer this, we need to consider what would be the average payoff. With probability 1/2, the player wins \$1; with probability 1/4, the player wins \$2; with probability 1/8, the player wins \$4, with probability $1/(2^n)$, the player wins $\$2^{n-1}$. The expected value is thus

$$E = \frac{1}{2} \times \$1 + \frac{1}{4} \times \$2 + \frac{1}{8} \times \$4 + \cdots \frac{1}{2^n} \times \$2^{n-1} + \cdots = \sum_{n=1}^{\infty} \frac{1}{2^n} \times \$2^{n-1} = \sum_{n=1}^{\infty} \frac{1}{2} = \infty$$

Assuming the game can continue as long as the coin toss results in heads and in particular that the casino has unlimited resources, this sum grows without bound and so the expected win for repeated play is an infinite amount of money. Considering nothing but the expectation value of the net change in one's monetary wealth, one should therefore play the game at any price if offered the opportunity. Contrary to this outcome, very few people are willing to pay large amounts of money to play this game. In fact, few of us would pay even \$25 to enter such a game (Martin, 2004). A hypothesized reason is that people perceive the risk associated with the game and consequently alter their behavior.

Bernoulli formalized this discrepancy between expected value and the behavior of individuals in terms of utility as the expected utility hypothesis: individuals make decisions with respect to investments in order to maximize expected utility (Sommer, 1954). The expected utility hypothesis is a description of human behavior.

In general, utility is the measure of satisfaction or value that the decision maker associates with each outcome. In practice, very often we establish a relationship between monetary outcomes and their utility, which provides the basis for formulating a maximum expected utility rule for decision making. This relationship is created in a form of utility function $u(\$)$ that assigns a numerical value of utility between 0 (least preferred) and 1 (most preferred) for each monetary outcome.

We use the following example for illustration. A person named Jeff is deciding to gamble in a casino. Jeff is given \$200 as a welcome gift. If Jeff chooses to play a game, in which there is 20% probability of winning \$5000 and 80% of losing \$1000, he will have to first pay the \$200 gift back to the casino in order to play the game. If he chooses not to play the game, he keeps the \$200.

A decision tree shown in Figure 16.9(a) depicted the situation, in which Jeff is choosing between Option A for not playing the game (walk away with \$200) and Option B for a chance to win \$5000 and risk losing \$1000. The expected values for options A and B are, respectively,

$$E(\text{A}) = \$200, \quad \text{and}$$

$$E(\text{B}) = 0.2(\$5000) + 0.8(-\$1000) = \$200$$

Thus, an expected value decision maker would be indifferent regarding A and B. In fact, more people may prefer Option A than B since A guarantees a \$200 gift; although B offers a chance of winning \$5000, the risk of losing \$1000 is too high to justify it.

Certainly, a higher probability of winning (p) or a different dollar amount of the welcome gift, designated as x, may alter Jeff's decision. If we generalize the decision illustrated in Figure 16.9(a) by not specifying a numerical value for p or x, the modified decision tree is shown in Figure 16.9(b).

Now, if for a given set of values for x and p, we conclude that options A and B are equally attractive, implying that their utilities are equal (i.e., $u(\text{A}) = u(\text{B})$) or in terms of p and x, we have

$$u(\$x) = p\,u(\$5000) + (1-p)\,u(-\$1000) \tag{16.13}$$

To construct the utility function for this problem, we examine a series of at least three decision problems. We consider a different value of x and ask what value of p is needed to make options A and B equally attractive—that is, $u(\text{A}) = u(\text{B})$.

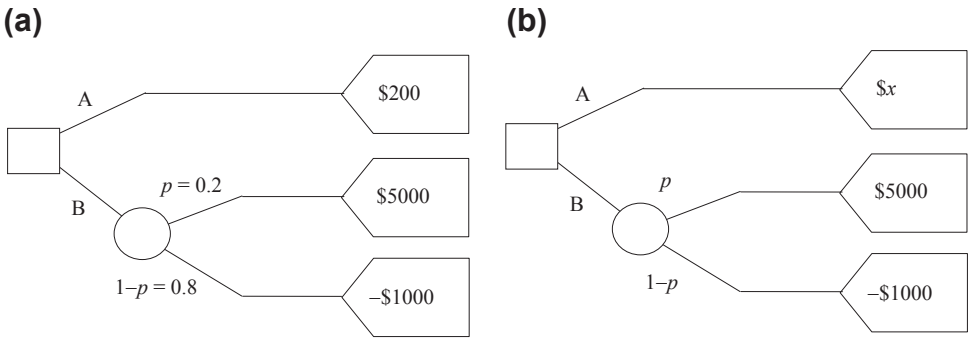


FIGURE 16.9

Decision tree for (a) outcomes A and B and (b) modified tree with p and x unspecified.

If $x = \$5000$, regardless of an individual's attitude toward risk, the only rational response is $p = 1$. That is, there is no gain or loss to either play or not play the game. It is then logical to assign utility 1 to $u(\$5000)$ as the most preferred outcome. On the other hand, if $x = -\$1000$, regardless of an individual's attitude toward risk, the only rational response is $p = 0$. That is, there is no gain or loss to either play or not play the game. It is then logical to assign utility 0 to $u(-\$1000)$ as the least preferred outcome.

Now, let us say the welcome gift given to Jeff is \$200. The question is how high the winning probability has to go in order to attract Jeff to play the game. Certainly, this percentage is different for different individuals because an individual's attitude toward risk is different. Let us say that when the winning percentage goes up to 40%, Jeff is indifferent between the two options; that is, he can go either way. To Jeff, the preference to the two options is identical. Therefore, $u(\$200)=0.4$. We may go over a similar exercise to obtain more utilities for different x values. Eventually, a utility function that represents Jeff's attitude toward risk can be constructed.

Instead of going over more exercises, we plot the dollar amounts and utilities of the three sets of values as three squares on graph formed by \$ and $u(\$)$ as abscissa and ordinate, respectively, as shown in Figure 16.10. If we assume the utility function is monotonic, a curve that passes through the three points shown in Figure 16.10 is defined as the utility function, which is concave. A concave utility function reflects the risk-averse (or risk-avoiding) nature of the decision maker, implying that the decision maker is more conservative. Note that the assumption of using a monotonic function is logical because a conservative decision maker will most likely stay conservative under circumstances in which the utility curve that represents the decision maker's attitude toward risk is always above the straight line, representing that of a risk-neutral decision maker. It takes a larger utility value of $u = 0.4$ for Jeff, a conservative decision maker, than a risk-neutral person ($u = 0.2$) to enter the game.

Once a utility function is constructed, like that in Figure 16.10, we are able to predict that when Jeff is offered a welcome gift of, for example \$2000, the casino must increase the probability of winning \$5000 to much higher than 50% in order to attract him to play the game. That is, for a given $\$x$, we can find p using the utility function, and vice versa. If the utility function accurately captures Jeff's attitude toward risk, Jeff may hire an agent or implement computer software to make a decision for him.

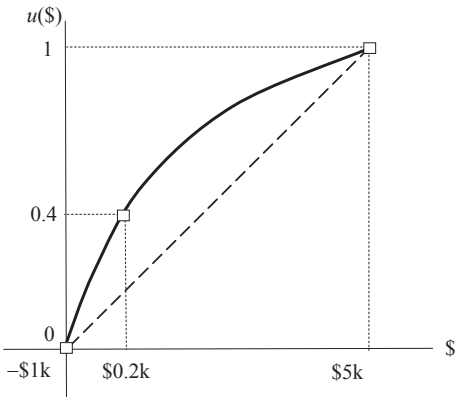


FIGURE 16.10

Utility function constructed for the example problem.

We mentioned that the concave utility function shown in Figure 16.10 reflects the risk-averse nature of the decision maker. A utility in the form of a straight line, as shown in Figure 16.11, indicates that the utility of any outcome is proportional to the dollar value of that outcome, which reflects those who make decisions following the expected value rule. We called these people risk-neutral decision makers. The convex curve shown in Figure 16.11 indicates the attitude of a risk-prone decision maker, who would choose a larger but uncertain benefit over a small, but certain, benefit.

16.4.5 CONSTRUCTION OF UTILITY FUNCTIONS

As discussed above, one way to construct a utility function is to acquire a client's (e.g., Jeff in the example of playing casino game) responses to a series of questions. In many cases, this approach may not be practical due to numerous factors, such as a client's availability. A more common approach to construct utility functions, which is not data intensive, is to build a mathematical model for the utility function by prescribing the parametric in a generic family of curves, which are monotonic. Many functions reveal the characteristics of monotonicity, such as the exponential function $y = e^x - 1$, $y = 1 - 2^{-x}$, and so forth. One such family of curves commonly adopted is

$$u(s) = \frac{1 - e^{-rs}}{1 - e^{-r}} \tag{16.14}$$

where s is a normalized form of the outcome x, defined as

$$s = \frac{x - x_{\text{worse}}}{x_{\text{best}} - x_{\text{worse}}} \tag{16.15}$$

where x_{best} and x_{worst} are the most preferred and the least preferred outcomes, respectively. When $x = x_{\text{worst}}$, $s = 0$; and $x = x_{\text{best}}$, $s = 1$. Also, when $r = 0$, $u(s) = s$, which is a straight line connecting (0,0) and (1,1), as shown in Figure 16.12, representing risk neutral. Note that $u(s) = s$ is obtained by applying L'Hôpital's rule, which we learned in calculus. When $r > 0$, the utility function is concave, representing risk-averse behavior; when $r < 0$, the utility function is convex, representing risk-prone behavior.

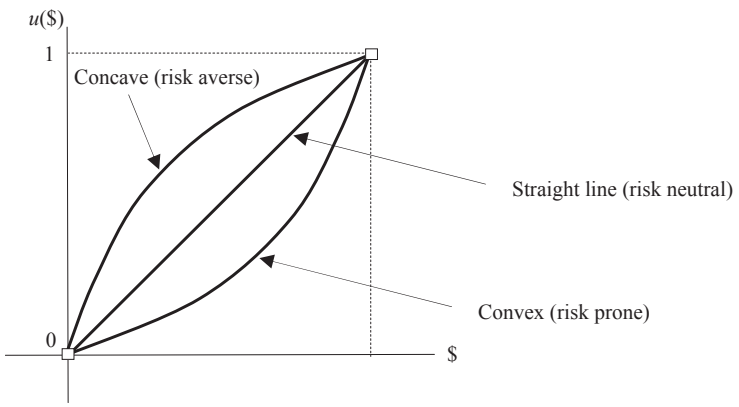


FIGURE 16.11

General form of risk-averse, risk-neutral, and risk-prone utility functions.

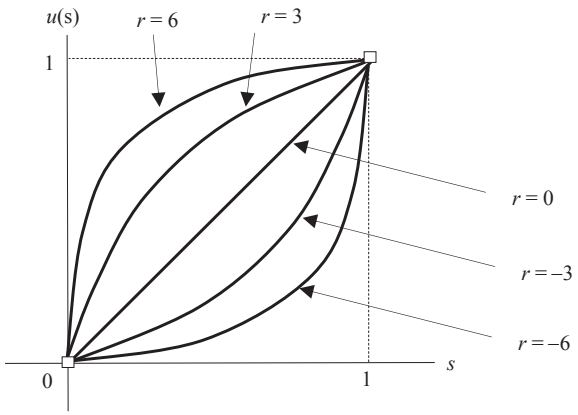


FIGURE 16.12

Mathematical model for the family of utility functions.

Next, we apply the decision-making rule based on the maximum expected utility to the car-buying example. Recall the decision tree shown in Figure 16.6(a). First, we identify the best and worst outcomes as $x_{\text{best}} = \$16{,}000$ and $x_{\text{worst}} = \$36{,}000$, respectively. From Eq. 16.15, we calculate the normalized parameter s corresponding to each monetary outcome, as shown in Table 16.7. We assume that the utility function defined in Eq. 16.14 with $r = 3$ accurately reflects the attitude toward risk of the buyer, indicating the car buyer is risk averse in spending money, like most of us. The utilities of individual outcomes can then be calculated using Eq. 16.14 and they are shown in Table 16.7. Using the utility values calculated in Table 16.7, the maximum expected utilities of buying new and used cars are, respectively,

$$u(N) = 0.937(0.818) + 0.0634(0) = 0.766$$

and

$$u(U) = 0.135(1) + 0.865(0.555) = 0.615$$

which shows that buying a new car is a better choice, incorporating the buyer's attitude toward risk.

16.4.6 MULTIATTRIBUTE UTILITY FUNCTIONS

To this point, our discussion on utility theory assumed a single attribute (or single design criteria, or single objective). In many situations, a decision maker may face multiple criteria. For example, a car buyer wants a car with a long expected lifespan and a low price. In general, expensive cars last longer, implying that the criteria may be in conflict for some cases. Moreover, lifespan and purchase price are different measures and are difficult to compare directly.

In general, utility functions can be constructed to convert numerical attribute scales to utility unit scales, which allow direct comparison of diverse measures. When decision making involves a single attribute, the utility function constructed for the attribute is called a single attribute utility (SAU)

Table 16.7 The Normalized Parameter and Utilities

x	*s*	*u*
$16,000	1	1.000
$26,000	0.5	0.818
$31,000	0.25	0.555
$36,000	0	0

function—for example, the function defined in Eq. 16.14. In this section, we introduce multiple attribute utility (MAU) functions and a structured methodology designed to handle the trade-offs among multiple objectives.

The mathematical combination of the SAU functions through certain scaling techniques yields a MAU function, which provides a utility function for the overall design with all attributes considered simultaneously. The scaling techniques reflect the decision maker's preferences on the attributes. For the formulation of the MAU function, additive and multiplicative formulations are commonly considered.

16.4.6.1 Additive MAU Functions

The main advantage of the additive utility function is its relative simplicity. The additive form, defined by Eq. 16.16, allows for no preference interactions among attributes. Thus, the change in utility caused by a problem in one attribute does not depend on whether there are any problems in other attributes.

For a set of outcomes x_1, x_2, …, x_m on their respective m attributes, its combined utility using additive form is computed as

$$u(x_1, x_2, \ldots, x_m) = k_1 u_1(x_1) + k_2 u_2(x_2) + \ldots k_m u_m(x_m) = \sum_{i=1}^{m} k_i u_i(x_i) \tag{16.16}$$

where $k_i > 0$ is the ith scaling factor and $\sum_{i=1}^{m} k_i = 1$, and u_i is the ith utility function. Note that k_i specifies the willingness of the decision maker to make trade-offs between different attributes. Recall that each utility function is normalized such that $u_i \in [0,1]$. Because the sum of all the scaling constants is 1, the value of MAU function defined in Eq. 16.16 is also between 0 and 1; that is, $u(x_1,x_2,\ldots,x_m) \in [0,1]$.

For example, a car buyer wants to buy a car with a long expected lifespan and a low price. The buyer narrowed down his or her choices to three alternatives: car A is a relatively expensive sedan with a reputation for longevity, car B is known for its reliability, and car C is a relatively inexpensive domestic automobile. The buyer has done some research and evaluated these three cars on both attributes, as shown in Table 16.8.

We set u_P and u_L for utilities of price and lifespan, respectively. Based on Table 16.8, we set $u_P(C) = u_P(\$8000) = 1$ for car C; $u_P(A) = u_P(\$17{,}000) = 0$ for car A. We assume the buyer is risk neutral on both price and lifespan; that is, $r = 0$ in the utility function defined in Eq. 16.14; that is, $u(s) = s$. Hence, $u_P(B) = u_P(\$10{,}000) = 0.778$ for car B. Similarly, we have $u_L(A) = u_L(12) = 1$; $u_L(C) = u_L(6) = 0$; hence, $u_L(B) = u_L(9) = 0.5$, as summarized in Table 16.9.

Table 16.8 Prices and Lifespans on the Three Candidate Cars

Attributes	Courses of Action		
	Car A	Car B	Car C
Price	$17,000	$10,000	$8000
Lifespan (years)	12	9	6

Table 16.9 Utility Values Set for the Two Attributes on the Three Candidate Cars

Attributes	Courses of Action		
	Car A	Car B	Car C
u_P	0	0.778	1
u_L	1	0.5	0

We assume that price is a lot more important factor than the lifespan of the car; therefore, we assign $k_P = 0.75$ and $k_L = 0.25$. Then, the utilities for the three alternatives are, respectively,

$$u(\text{A}) = k_P\, u_P(\text{A}) + k_L\, u_L(\text{A}) = 0.75\ (0) + 0.25\ (1) = 0.25$$

$$u(\text{B}) = k_P\, u_P(\text{B}) + k_L\, u_L(\text{B}) = 0.75\ (0.778) + 0.25\ (0.5) = 0.709$$

$$u(\text{C}) = k_P\, u_P(\text{C}) + k_L\, u_L(\text{C}) = 0.75\ (1) + 0.25\ (0) = 0.75$$

Example 16.1 offers more detailed illustration.

EXAMPLE 16.1

A different car buyer wants to buy a car with a long expected lifespan and a low price and has narrowed his or her choices down to the same three alternatives cars A, B, and C, with price and lifespan information listed in Table 16.8. The car buyer is risk averse in purchase price and is neutral in lifespan. We assume a utility function defined in Eq. 16.14 with $r = 3$, which characterizes the buyer's attitude toward risk. Calculate the combined utilities for the three alternatives using the additive form. We assume that price is equally as important as the lifespan of the car; therefore, $k_P = 0.5$ and $k_L = 0.5$.

Solutions

Because the buyer is risk neutral on lifespan, we have $u_L(\text{A}) = u_L(12) = 1$; $u_L(\text{C}) = u_L(6) = 0$; hence, $u_L(\text{B}) = u_L(9) = 0.5$. The buyer is risk averse with $r = 3$. From Eq. 16.15, we have

$$s_P = \frac{x - x_{\text{worse}}}{x_{\text{best}} - x_{\text{worse}}} = \frac{x - \$17,000}{\$8000 - \$17,000} = -\frac{x - \$17,000}{\$9000}$$

Hence, $s_P(\text{B}) = s_P(\$10{,}000) = 0.778$. Then, from the utility function defined in Eq. 16.14 with $r = 3$, we have

$$u_P(\text{B}) = u(s_P(\text{B})) = \frac{1 - e^{-rs_P(\text{B})}}{1 - e^{-r}} = \frac{1 - e^{-3(0.778)}}{1 - e^{-3}} = 0.950$$

Continued

EXAMPLE 16.1—cont'd

Then, the utilities for the three alternatives are, respectively,

$$u(\mathrm{A}) = k_{\mathrm{P}}\, u_{\mathrm{P}}(\mathrm{A}) + k_{\mathrm{L}}\, u_{\mathrm{L}}(\mathrm{A}) = 0.5\ (0) + 0.5\ (1) = 0.5$$

$$u(\mathrm{B}) = k_{\mathrm{P}}\, u_{\mathrm{P}}\ (\mathrm{B}) + k_{\mathrm{L}}\, u_{\mathrm{L}}\ (\mathrm{B}) = 0.5\ (0.950) + 0.5\ (0.5) = 0.725$$

$$u(\mathrm{C}) = k_{\mathrm{P}}\, u_{\mathrm{P}}(\mathrm{C}) + k_{\mathrm{L}}\, u_{\mathrm{L}}(\mathrm{C}) = 0.5\ (1) + 0.5\ (0) = 0.5$$

16.4.6.2 Multiplicative MAU Functions

The other commonly employed MAU functions are multiplicative, which support preference interactions among attributes. A multiplicative MAU function is defined in Eq. 16.17:

$$u(x_1, x_2, \ldots, x_m) = \frac{1}{K}\left\{\prod_{i=1}^{m}[1 + Kk_i u_i(x_i)] - 1\right\} \tag{16.17}$$

K is a new parameter called the composite scaling constant, which satisfies the following equation:

$$1 + K = \prod_{i=1}^{m}(1 + Kk_i) \tag{16.18}$$

If the scaling factor is $K = 0$, indicating no interacting of attribute preference, this formulation is equivalent to its additive form, assuming the sum of all the scaling factors k_i is 1; that is, $\sum_{i=1}^{m} k_i = 1$.

It is shown in Hyman (2003) and Shtub et al. (1994) that if $\sum_{i=1}^{m} k_i > 1$, then $-1 < K < 0$, and if $\sum_{i=1}^{m} k_i < 1$, then $K > 0$. Examples 16.2 and 16.3 provide a few more details on the multiplicative MAU function.

EXAMPLE 16.2

For $m = 2$ and $k_1 + k_2 = 1$, $k_1 > 0$ and $k_2 > 0$, show that the scaling factor defined in Eq. 16.18 is $K = 0$, and the multiplicative MAU function u defined in Eq. 16.17 becomes a linear MAU function.

Solutions

From Eq. 16.18, we have

$$1 + K = \prod_{i=1}^{2}(1 + Kk_i) = (1 + Kk_1)(1 + Kk_2) = 1 + K(k_1 + k_2) + K^2 k_1 k_2$$

Because $k_1 + k_2 = 1$, the above equation becomes $1 + K = 1 + K + K^2 k_1 k_2$. Because $k_1 > 0$ and $k_2 > 0$, we have $K = 0$. Hence,

$$u(x_1, x_2, \ldots, x_m) = \frac{1}{K}\left\{\prod_{i=1}^{m}[1 + Kk_i u_i(x_i)] - 1\right\} = \frac{1}{K}\{(1 + Kk_1u_1)(1 + Kk_2u_2) - 1\}$$

$$= \frac{1}{K}\{K(k_1u_1 + k_2u_2) + K^2 k_1u_1k_2u_2\} = k_1u_1 + k_2u_2$$

EXAMPLE 16.3

Continue with Example 16.1 but use a multiplicative MAU function with $k_P = 0.5$ and $k_L = 0.25$.

Solutions

First, from Eq. 16.18, we have

$$1 + K = \prod_{i=1}^{2}(1 + Kk_i) = 1 + K(k_P + k_L) + K^2 k_P k_L = 1 + 0.75K + 0.125K^2$$

Hence $K = 2$.

Then, the utilities for the three alternatives are, respectively,

$$u(\mathrm{A}) = \frac{1}{K}\{(1 + Kk_P u_P(\mathrm{A}))(1 + Kk_L u_L(\mathrm{A})) - 1\} = \frac{1}{2}\{(1 + 2(0.5)(0))(1 + 2(0.25)(1)) - 1\} = 0.25$$

$$u(\mathrm{B}) = \frac{1}{K}\{(1 + Kk_P u_P(\mathrm{B}))(1 + Kk_L u_L(\mathrm{B})) - 1\} = \frac{1}{2}\{(1 + 2(0.5)(0.950))(1 + 2(0.25)(0.5)) - 1\} = 0.719$$

$$u(\mathrm{C}) = \frac{1}{K}\{(1 + Kk_P u_P(\mathrm{C}))(1 + Kk_L u_L(\mathrm{C})) - 1\} = \frac{1}{2}\{(1 + 2(0.5)(1))(1 + 2(0.25)(0)) - 1\} = 0.5$$

16.5 GAME THEORY

The design theory we discussed so far exclusively focused on a situation in which a single decision maker needs to find a best possible decision among alternatives. A decision needed to be made by one decision maker to produce maximum payoffs in single or multiple attributes (or criteria) under a set of constraints. In many situations, however, the payoff of a decision made by an individual depends not only on what he or she does but also on the outcome of the decisions or choices that other individuals make. In engineering design, design decisions are often not made by a single designer or a single design team. Instead, multiple designers or multiple design teams are working on the product design and are involved in design decision making, with each designer or team being responsible for one or more design objectives and/or subsystems. For example, a structural engineer focuses on maximizing the strength and durability of a load-bearing component, while the goal of a manufacturing engineer is to produce the component with least cost in less time. Design decisions made by the structural engineer in determining the geometric shape of the component may affect the cost and time of manufacturing the component and vice versa.

In practice, some of the design decisions are made simultaneously at a specific time of a design phase, and some are made in sequence throughout the design process. With several designers (or design teams) each with his or her own objectives, the nature of the design decisions can take several paths and the overall design may not be desired. This is because a single designer or team can theoretically do better and his or her decision could dominate, hence largely determining the performance of the overall product, which may or may not be desired.

Game theory applied to this situation provides a method for understanding and perhaps guiding the design decision making. Game theory is a study of strategic decision making. More formally, it is the study of mathematical models of conflict and cooperation between intelligent, rational decision makers. As such, game theory could serve as an important and effective management tool for use in the situations of multiple designers or multiple design teams that are decentralized.

Game theory is mainly used in economics, political science, and psychology, as well as logic and biology. It was recently explored for engineering design. Our purpose of discussing game theory and later employing the theory as a design tool in Section 16.6.2 is not to offer a complete solution for design decision making but to present a plausible idea that has been explored for its feasibility for support of engineering design. As of now, applying game theory as a design tool is still an open research topic that requires much effort to convert the theory into practical tools in the near future.

In the following, we first discuss in Section 16.5.1 the elements of a game and the strategy to determine an equilibrium (or a rational solution) to the game; that is, a stable state in which either one outcome occurs or a set of outcomes occur with known probability. Then, in Section 16.5.2, we present a two-person matrix game and discuss Nash equilibrium for the game. The information presented in these two subsections should provide enough information for readers to proceed with two other kinds of games, sequential games and cooperative games, which are discussed in Sections 16.5.3 and 16.5.4, respectively. To minimize the complexity in mathematical discussion, we assume for most cases two-player games. Readers are referred to the literature (e.g., Aliprantis and Chakrabarti, 2006; Hazelrigg, 1996) for a more thorough and in-depth discussion of the subject.

16.5.1 ELEMENTS OF A GAME

We start by introducing a well-known game in the literature: the prisoner's dilemma game or prisoner's game. Thereafter, we introduce basic elements in a game by referring to the prisoner's game as an example. With this basic understanding, we proceed with formulating a game mathematically.

In the prisoner's game, two criminals are apprehended by the police, who have strong evidence that they are guilty of the crime, but only weak evidence that they are guilty of a second. Without a confession by one criminal or the other, a conviction for the second crime cannot be obtained. Upon capture, the criminals are immediately separated for interrogation. They are both told that existing evidence will convict them of the first crime, and for that they will each serve 5 years in jail. However, they are offered the opportunity to confess to the second crime. If one confesses and the other one does not, the confessor will have his sentence reduced to 3 years, while the other will have his sentence lengthened to 10 years. But if both confess, both their sentences will be lengthened to 8 years. Certainly, the criminals desire to serve less time in jail. Therefore, a payoff table with negative of the time served in jail is constructed in Table 16.10. The first numeric figure in the pair denotes the payoff for prisoner A and the second for prisoner B.

If the prisoners could hold each other to binding agreements, both would be better off if neither confessed; in this case, each gets 5 years. This is a so-called cooperative game because both players cooperate to maximize their collective payoffs. Acting alone, the better strategy for each is to confess. This is a so-called noncooperative game. If prisoner A chooses to confess, B gets 10 years if he

Table 16.10 A Payoff Table for the Prisoner's Dilemma Game

		Prisoner B	
		Confess	**Withhold**
Prisoner A	Confess	−8, −8	−3, −10
	Withhold	−10, −3	−5, −5

withholds or 8 years if he confesses. If A chooses to withhold, B gets 5 years if he withholds and only 3 years if he confesses. In short, no matter what A does, B is better off by confessing. We say that the strategy of confessing is a strictly dominant strategy for B. The same applies to A. Thus, both prisoners confess, and both are worse off for their actions.

In the absence of any communication or any coordination scheme, rational players are expected to play their strictly dominant strategies because a strictly dominant strategy gives a player an unequivocally higher payoff. A solution to the prisoner's dilemma game can therefore end up being (confess, confess). This is the solution using strictly dominant strategies. The solution (confess, confess) is called the Nash equilibrium, which will be discussed further shortly.

In this prisoner's game—and any game in general—the following elements are present:

1. There are two (or more) participants (or players). In this example, there are two criminals, A and B.
2. Each player has a set of alternative choices, and a player can take action on the available choices. In this example, each criminal can choose to confess or withhold. We denote by a_i a choice that player i can make. We refer to the set of choices available to the player i as player i's action set, $A_i = \{a_i\}$.
3. Each player develops a strategy to play the game. A strategy s_i is a rule to tell player i which available actions to choose, subject to available information, each time he or she takes an action. In this example, the strategy set for criminal B that maximizes payoffs (provided that information of A's action is available) includes: if A confesses, B confesses; if A withholds, B confesses. The strategies available to player i are referred to as the player's strategy set or strategy space $S_i = \{s_i\}$. For all n players in a game, we define the strategy combination or strategy profile $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$. For the prisoner's game, the strategy combination is $\boldsymbol{s} = (s_A, s_B)$.
4. For each outcome, there is a payoff that each player gets. In the case of the prisoner's game, the payoffs are given by the pairs (a, b) for each outcome, as shown in Table 16.10.

The above are the essential ingredients that constitute what is called a game in strategic form (or normal form). Note that, as seen in (2) and (3), we use action and strategy interchangeably. In general, a strategic form game consists of a set of players; for each player, there is a strategy set; and for each outcome (or strategy combination) of the game, there is a payoff for each player. When a game is presented in normal form, it is presumed that each player acts simultaneously or, at least, without knowing the actions of the other. If players have some information about the choices of other players, the game is usually presented in extensive form (or tree form). We discuss the extensive form of a sequential game in Section 16.5.3.

Next, we formulate the game mathematically and discuss solution techniques. We focus on two-player games in a matrix form first.

16.5.2 TWO-PERSON MATRIX GAMES

A matrix game, such as the prisoner's dilemma game, is a two-player game such that:

1. Player A has a finite strategy set S_A with m elements; that is, $S_A = \{s_{A1}, s_{A2}, \ldots, s_{Am}\}$. In the prisoner's dilemma game, $S_A = \{s_{A1} = \text{confess}, s_{A2} = \text{withhold}\}$.
2. Player B has a finite strategy set S_B with n elements; that is, $S_B = \{s_{B1}, s_{B2}, \ldots, s_{Bn}\}$. In the prisoner's dilemma game, $S_B = \{s_{B1} = \text{confess}, s_{B2} = \text{withhold}\}$.
3. Denoting s_A an element of set S_A (i.e., $s_A \in S_A$) and s_B an element of set S_B (i.e., $s_B \in S_B$), the payoffs of the players are measured by utility functions $u_A(s_A,s_B) \in R$ and $u_B(s_A,s_B) \in R$ (R is a real number) of the outcomes, in which $\boldsymbol{s} = (s_A,s_B) \in S_A \times S_B$. For example, in the prisoner's

Table 16.11 The Payoff Table of the Two-Person Matrix Game

		Player B			
Player A	Strategy	s_{B1}	s_{B2}	...	s_{Bn}
	s_{A1}	(a_{11}, b_{11})	(a_{12}, b_{12})	...	(a_{1n}, b_{1n})
	s_{A2}	(a_{21}, b_{21})	(a_{22}, b_{22})	...	(a_{2n}, b_{2n})
	...	...	...	...	...
	s_{Am}	(a_{m1}, b_{m1})	(a_{m2}, b_{m2})	...	(a_{mn}, b_{mn})

game, u_A (s_{A1} = confess, s_{B2} = withhold) = −3, and u_B (s_{A1} = confess, s_{B2} = withhold) = −10. Again, $\boldsymbol{s} = (s_A, s_B)$ is called a strategy combination or a strategy profile.

Mathematically, the matrix game formulated above can be described as follows: At a certain time, player A chooses a strategy $s_A \in S_A$. Simultaneously and independently, player B chooses a strategy $s_B \in S_B$. Once this is done or once a strategy profile $\boldsymbol{s} = (s_A, s_B)$ is determined, each player receives the respective payoff $a_{ij} = u_A(s_{Ai}, s_{Bj})$ and $b_{ij} = u_B(s_{Ai}, s_{Bj})$. The payoffs can be arranged in the form of an $m \times n$ matrix shown in Table 16.11.

The idea of a solution of a game is usually identified by the concept of the Nash equilibrium, defined next.

A pair of strategies $(s_A^*, s_B^*) \in S_A \times S_B$ is a Nash equilibrium of a matrix game if

$$1.\quad u_A(s_A^*, s_B^*) \geq u_A(s_A, s_B^*) \text{ for each } s_A \in S_A, \quad \text{and} \tag{16.19}$$

$$2.\quad u_B(s_A^*, s_B^*) \geq u_B(s_A^*, s_B) \text{ for each } s_B \in S_B \tag{16.20}$$

in which the strictly dominant strategies are followed. In fact, a Nash equilibrium is an outcome of the game from which none of the players have an incentive to deviate. This is because it is optimal for a player to choose the Nash equilibrium strategy. In this sense, a Nash equilibrium has a property that is self-enforcing. Example 16.4 provides more illustration.

EXAMPLE 16.4

Verify that in the prisoner's dilemma example, the pair of strategies (confess, confess) is a Nash equilibrium and (withhold, withhold) is not.

Solutions

In the prisoner's dilemma example, the strategy sets for players A and B are, respectively, $S_A = \{s_{A1} = \text{confess}, s_{A2} = \text{withhold}\}$ and $S_B = \{s_{B1} = \text{confess}, s_{B2} = \text{withhold}\}$, and $m = n = 2$. The pair of strategies $(s_A^* = s_{A1} = \text{confess},\ s_B^* = s_{B1} = \text{confess})$ is a Nash equilibrium because, for player A.

1. $u_A(s_A^*, s_B^*) \geq u_A(s_A, s_B^*)$ for each $s_A \in S_A$; that is, $u_A(s_A^* = s_{A1} = \text{confess},\ s_B^* = s_{B1} = \text{confess}) = -8$ and $u_A(s_{A1} = \text{confess},\ s_B^* = s_{B1} = \text{confess}) = -8$ and $u_A(s_{A2} = \text{withhold},\ s_B^* = s_{B1} = \text{confess}) = -10$.

For player B,

2. $u_B(s_A^*, s_B^*) \geq u_B(s_A^*, s_B)$ for each $s_B \in S_B$; that is, $u_B(s_A^* = s_{A1} = \text{confess},\ s_B^* = s_{B1} = \text{confess}) = -8$ and $u_B(s_A^* = \text{confess},\ s_B = s_{B1} = \text{confess}) = -8$ and $u_B(s_A^* = \text{confess},\ s_B = s_{B2} = \text{withhold}) = -10$.

Note that if both players withhold, each receives a 5-year sentence, which seems to be a better solution to the situation. Is it a Nash equilibrium? Let us take a look. For player A,

1. $u_A(s_A^* = s_{A1} = \text{withhold},\ s_B^* = s_{B1} = \text{withhold}) = -5$, and $u_A(s_{A1} = \text{withhold},\ s_B^* = s_{B1} = \text{withhold}) = -5$ and $u_A(s_{A2} = \text{confess},\ s_B^* = s_{B1} = \text{withhold}) = -3$. Therefore, $u_A(s_A^*, s_B^*) \geq u_A(s_A, s_B^*)$ does not hold.

EXAMPLE 16.4—cont'd

For player B,

2. $u_B(s_A^* = s_{A1} = \text{withhold},\ s_B^* = s_{B1} = \text{withhold}) = -5$, and $u_B(s_A^* = \text{withhold},\ s_B = s_{B1} = \text{withhold}) = -5$ and $u_B(s_A^* = \text{withhold},\ s_B = s_{B2} = \text{confess}) = -3$. Therefore, $u_B(s_A^*, s_B^*) \geq u_B(s_A^*, s_B)$ does not hold.

According to Eqs 16.19 and 16.20, (withhold, withhold) is not a Nash equilibrium and is not self-enforcing. This is because that there is no binding agreement between the prisoners, and a player receives a longer sentence (10 years) if he chooses to withhold while the other player confesses.

As shown in Example 16.4, finding the Nash equilibrium by verifying Eqs 16.19 and 16.20 for all possible strategies may not be practical, especially when there are more than two players and each player is given a large strategy set. There are numerous ways to find the Nash equilibrium. In this subsection, we present the approach of a necessary condition learned in calculus followed by Example 16.5 for illustration. Later, in Section 16.6.2, we employ a graphical approach for a pressure vessel design problem.

We assume the strategy sets are open intervals of real numbers and the payoff functions are differentiable. In this case, a necessary condition for a strategy set $(s_1^*, s_2^*, \ldots, s_n^*)$ of an n-player game to be a Nash equilibrium of the game is

$$\frac{\partial u_i(s_1^*, s_2^*, \ldots, s_n^*)}{\partial s_i} = 0, \quad i = 1, n \tag{16.21}$$

When the system of equations of Eq. 16.21 has a unique solution, then it is the only possible Nash equilibrium of the game. In many cases, the Nash equilibrium is not unique. In some situations, other factors need to be considered to verify that the solution is the Nash equilibrium of the game—for example, second derivatives of the payoff functions with respect to the strategy set.

The next example is extracted from Aliprantis and Chakrabarti (2006) and is called the Cournot duopoly model, initially analyzed by French mathematician Augustin Cournot.

EXAMPLE 16.5

Two firms, firm 1 and firm 2, produce an identical product of amounts q_1 and q_2, respectively. The price per unit of the product in the market is determined by $p(q) = A - q$, where $q = q_1 + q_2$, and A is a fixed number. The price equation shows that the per-unit price of the product is decreasing with an increasing total production amount $q = q_1 + q_2$. Assume that the total cost for firm i of producing the output q_i is $C_i = c_i q_i$, where c_i is cost per unit, a positive number. What are the optimal outputs q_1 and q_2 that each firm should produce in order to maximize profit? Note that the profit of each firm depends on the output of the other firm. They choose their production quantities independently and simultaneously. There is no communication or binding agreement between the firms in regard to the production quantities.

Solutions

This problem can be modeled as a two-player matrix game that is noncooperative. It is reasonable to think of the Nash equilibrium as a self-enforced solution. We find the Nash equilibrium of the game using the first-order derivative test of Eq. 16.21.

We first formulate an equation describing profit as function of quantities. For firms 1 and 2, we have, respectively,

$$u_1(q_1, q_2) = p(q)\, q_1 - c_1 q_1 = (A - q_1 - q_2)\, q_1 - c_1 q_1 = -(q_1)^2 + (A - q_2 - c_1)\, q_1 \tag{16.22}$$

and

$$u_2(q_1, q_2) = p(q)\, q_2 - c_2 q_2 = (A - q_1 - q_2)\, q_2 - c_2 q_2 = -(q_2)^2 + (A - q_1 - c_2)\, q_2 \tag{16.23}$$

Continued

EXAMPLE 16.5—cont'd

Taking derivatives of Eq. 16.22 with respect to q_1 and Eq. 16.23 with respect to q_2, we have

$$\frac{\partial u_1(q_1, q_2)}{\partial q_1} = -2q_1 - q_2 + A - c_1 = 0 \tag{16.24}$$

$$\frac{\partial u_2(q_1, q_2)}{\partial q_2} = -q_1 - 2q_2 + A - c_2 = 0 \tag{16.25}$$

Solving Eqs 16.24 and 16.25, we have

$$q_1^* = \frac{A + c_2 - 2c_1}{3} \tag{16.26}$$

$$q_2^* = \frac{A + c_1 - 2c_2}{3} \tag{16.27}$$

Note that if $A \geq c_1 + c_2$, then $q_1^* > 0$ and $q_2^* > 0$, implying that both firms produce a positive output at the Nash equilibrium.

If we assume that $A = 200$ and $c_1 = c_2 = 2$, then from Eq. 16.26, $q_1^* = 66$, and from Eq. 16.27, $q_2^* = 66$. Then, from Eqs 16.22 and 16.23, $u_1 = u_2 = 4356$. Is this the best payoff for each firm?

16.5.3 SEQUENTIAL GAMES

A sequential game involves multiple players who do not make decisions simultaneously, and one player's decision affects the outcomes and decisions of other players. A sequential game is represented by a game tree (also called the extensive form) with players moving sequentially. We assume information is perfectly known to a player at the time of decision making.

Consider a simple two-person sequential game of two stages with perfect information, whose game tree is shown in Figure 16.13. Here, each vertex (or node) represents a point of choice for a player. The player is specified by an alphabet listed by the vertex. The arrow out of the vertex represents a possible action for that player. The payoffs are specified at the terminal nodes placed at bottom of the tree. Two-stage implies that decisions are made in two stages, one by each player.

In the game shown in Figure 16.13, there are two players. Player A moves first and chooses either L1 or R1. Player B sees player A's move and then chooses L2 or R2, or L3 or R3 depending on player A's move. Suppose that player A chooses L1; then, node 2 is reached; if player B chooses L2, in which terminal node 4 is reached, then player A gets 2 and player B gets 1. If player A chooses R1, then node 3 is reached and if player B chooses R3, then node 7 is reached; then, player A gets 1 and player B gets 0. Now, we are ready to define a sequential game of two players.

A tree T is said to be a two-player game tree if the following are true:

1. Each nonterminal node of the tree is owned by exactly one player. For example, in Figure 16.13, node 1 is owned by player A, and nodes 2 and 3 are owned by player B.

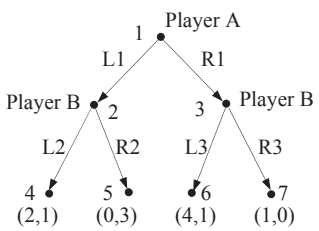


FIGURE 16.13

A game tree for a sequential game.

2. At each terminal node v of the tree, a payoff vector is assigned; that is, $\mathrm{u}(v) = (u_{\mathrm{A}}(v), u_{\mathrm{B}}(v))$. For example, at terminal node 4, $\mathrm{u}(4) = (2,1)$, as shown in Figure 16.13.

The above definition for a sequential game of two players can be easily extended to that of n players.

A strategy for a player i in a sequential game consists of the choices that the player is going to make at the nodes he or she owns. As illustrated in Figure 16.13, a strategy for a player in a sequential game is a complete plan of how to play the game, prescribing the choices at every node owned by the player. For example, in Figure 16.13, at node 2, two choices are available for player B: L2 and R2. In other words, a player's strategy will indicate the choices that the player has planned to make a priori (i.e., before the game starts). A strategy profile for a two-person sequential game is then represented as $\mathrm{s} = (s_{\mathrm{A}}, s_{\mathrm{B}})$, where each s_{A} and s_{B} is a strategy for players A and B, respectively. For example, as shown in Figure 16.13, S_{A} can be L1 or R1, and S_{B} is a function from {node 2, node 3} to $\{s_{\mathrm{B21}} = \mathrm{L2}, s_{\mathrm{B22}} = \mathrm{R2}; s_{\mathrm{B31}} = \mathrm{L3}, s_{\mathrm{B32}} = \mathrm{R3}\}$ with the feasibility restriction that, from node 2, player B can only choose L2 or R2 with a similar restriction on choices from node 3. In other words, the choice of player B depends on the choice made by player A according to the prescribed game tree. Note that a strategy profile uniquely determines the terminal node v that is reached. The payoff (or utility) of each player is a function u_{A} and u_{B} of the strategy profile $(s_{\mathrm{A}}, s_{\mathrm{B}})$.

A solution of a sequential game is also understood to be a Nash equilibrium. In a two-player sequential game, a strategy profile $(s_{\mathrm{A}}^*, s_{\mathrm{B}}^*)$ is said to be a Nash equilibrium if, for each player, we have

$$1.\quad u_{\mathrm{A}}(s_{\mathrm{A}}^*, s_{\mathrm{B}}^*) \geq u_{\mathrm{A}}(s_{\mathrm{A}}, s_{\mathrm{B}}^*) \quad \text{for each } s_{\mathrm{A}} \in S_{\mathrm{A}}, \quad \text{and} \tag{16.28}$$

$$2.\quad u_{\mathrm{B}}(s_{\mathrm{A}}^*, s_{\mathrm{B}}^*) \geq u_{\mathrm{B}}(s_{\mathrm{A}}^*, s_{\mathrm{B}}) \quad \text{for each } s_{\mathrm{B}} \in S_{\mathrm{B}} \tag{16.29}$$

In other words, a Nash equilibrium is a strategy profile $(s_{\mathrm{A}}^*, s_{\mathrm{B}}^*)$ such that player A cannot improve his payoff by changing his strategy if the player B does not change, and vice versa. Note that Eqs 16.28 and 16.29 are identical to those of Eqs 16.19 and 16.20 in a strategy form game.

A backward induction method is often employed to solve for the Nash equilibrium of a sequential game. Backward induction is the process of reasoning backward in time, from the end of a problem or situation, to determine a sequence of optimal actions. It proceeds by first considering the last time a decision might be made and choosing what to do in any situation at that time. Using this information, one can then determine what to do at the second-to-last time of decision. This process continues backward until one has determined the best action for every possible situation at every point in time. We illustrate the use of the method in the following example (Example 16.6).

EXAMPLE 16.6

Find the Nash equilibrium for the sequential game shown in Figure 16.13.

Solutions

We use the backward induction and conditions shown in Eqs 16.28 and 16.29 to find the Nash equilibrium.

We first assume $s_{\mathrm{A}}^* = s_{\mathrm{A1}} = \mathrm{L1}$, then $s_{\mathrm{B}}^* = s_{\mathrm{B2}} = \mathrm{R2}$ because $u_{\mathrm{B}}(\mathrm{L1}, \mathrm{R2}) = 3 \geq \{u_{\mathrm{B}}(\mathrm{L1}, \mathrm{R2}) = 3, u_{\mathrm{B}}(\mathrm{L1}, \mathrm{L2}) = 1\}$. If $s_{\mathrm{A}}^* = s_{\mathrm{A2}} = \mathrm{R1}$, then $s_{\mathrm{B}}^* = s_{\mathrm{B3}} = \mathrm{L3}$ because $u_{\mathrm{B}}(\mathrm{R1}, \mathrm{L3}) = 1 \geq \{u_{\mathrm{B}}(\mathrm{R1}, \mathrm{L3}) = 1, u_{\mathrm{B}}(\mathrm{R1}, \mathrm{R3}) = 0\}$.

Now, we go backward and use Eq. 16.28 to find s_{A}^* from the two strategies (L1, R2) and (R1, L3). We found that $s_{\mathrm{A}}^* = s_{\mathrm{A2}} = \mathrm{R1}$ because $u_{\mathrm{A}}(\mathrm{R1}, \mathrm{L3}) = 4 \geq \{u_{\mathrm{A}}(\mathrm{R1}, \mathrm{L3}) = 4, u_{\mathrm{A}}(\mathrm{L1}, \mathrm{R2}) = 0\}$. Hence the Nash equilibrium is found at $(s_{\mathrm{A}}^*, s_{\mathrm{B}}^*) = (\mathrm{R1}, \mathrm{L3})$—that is, the path $1 \rightarrow 3 \rightarrow 6$. If player B does not change its strategy $(s_{\mathrm{B}}^* = \mathrm{L3})$, player A cannot improve its payoff by changing its strategy—for example, from R1 to L1. Similarly, if player A does not change its strategy $(s_{\mathrm{A}}^* = \mathrm{R1})$, player B cannot improve its payoff by changing its strategy—for example, from L3 to R2.

The next example (Example 16.7) is again extracted from Aliprantis and Chakrabarti (2006), called the Stackelberg duopoly model, which is slightly modified from the example of the Cournot duopoly model to illustrate a few more details of the sequential game.

EXAMPLE 16.7

Continue with Example 16.5, except that firm 1 chooses a production quantity $q_1 \geq 0$. Firm 2 observes q_1 and then chooses its own production quantity q_2. Find the optimal production quantities q_1 and q_2 each firm should produce in order to maximize profit.

Solutions

This problem can be formulated as a two-person sequential game of two stages and with perfect information. We use backward induction again for this problem. We first find the output q_2^* of firm 2 that maximizes firm 2's profit given the output q_1 of firm 1. That is, $q_2^* = q_2^*(q_1)$. The profit of firm 2 can be formulated as

$$u_2(q_1, q_2^*) = \max_{q_2 \geq 0} \; u_2(q_1, q_2) = \max_{q_2 \geq 0} \; \left[-(q_2)^2 + (A - q_1 - c_2)q_2 \right] \tag{16.30}$$

Taking the first and second derivatives of Eq. 16.30 with respect to q_2, we have

$$\frac{\partial u_2}{\partial q_2} = -2q_2 + (A - q_1 - c_2) \tag{16.31}$$

and

$$\frac{\partial^2 u_2}{\partial q_2^2} = -2 < 0 \tag{16.32}$$

Solving for q_2 from Eq. 16.31, we have

$$q_2^* = q_2^*(q_1) = \frac{A - q_1 - c_2}{2} \tag{16.33}$$

which gives maximum profit u_2 for firm 2, provided $q_1 < A - c_2$ (so that $q_2^* > 0$).

Firm 1 should now anticipate that firm 2 will choose q_2^* if firm 1 chooses q_1. Therefore, firm 1 will want to choose q_1 to maximize its profit, defined as

$$u_1(q_1, q_2^*) = -(q_1)^2 + q_1(A - q_2^* - c_1) = -(q_1)^2 + q_1\left(A - \frac{A - q_1 - c_2}{2} - c_1\right) = \frac{1}{2}\left[-(q_1)^2 + q_1(A + c_2 - 2c_1) \right] \tag{16.34}$$

subject to $q_1 \geq 0$. Taking again the first and second derivatives of Eq. 16.34 with respect to q_1, we have

$$\frac{\partial u_1}{\partial q_1} = -q_1 + \frac{A + c_2 - 2c_1}{2} \tag{16.35}$$

and

$$\frac{\partial^2 u_1}{\partial q_1^2} = -1 < 0 \tag{16.36}$$

Therefore, from Eq. 16.35, we have

$$q_1^* = \frac{A + c_2 - 2c_1}{2} \tag{16.37}$$

which gives the maximum profit u_1 for firm 1. Substituting Eq. 16.37 to Eq. 16.33, we have

$$q_2^* = \frac{A - \frac{A + c_2 - 2c_1}{2} - c_2}{2} = \frac{A + 2c_1 - 3c_2}{4} \tag{16.38}$$

If we assume $A = 200$ and $c_1 = c_2 = 2$, then from Eq. 16.37, $q_1^* = 99$. From Eq. 16.38, $q_2^* = 49.5$. Then, $u_1 = 4900.5$ and $u_2 = 2450.25$.

16.5.4 COOPERATIVE GAMES

Now we go back to the prisoner's game. We have observed that both criminals in the prisoner's game are better off if they both withhold. This is the basis for one solution concept in cooperative games.

First, we define a criterion to rank outcomes from the point of view of the group of players as a whole. We can say that one outcome is better than another if at least one player is better off and no one is worse off. For example, in the prisoner's game (withhold, withhold) is better than (confess, confess). This is called the Pareto criterion, after the Italian economist and mechanical engineer, Vilfredo Pareto. If an outcome cannot be improved, then we say that the outcome is Pareto optimal—that is, optimal in terms of the Pareto criterion. In the prisoner's game (withhold, withhold) is Pareto optimal because no one can be made better off without making the other prisoner worse off. In fact, (confess, withhold) is also Pareto optimal because neither player is better off without making the other player worse off.

Mathematically, the Pareto optimal for a two-person game can be defined as follows:

$$\text{A strategy profile } \mathrm{s}^* = \left(s_{\mathrm{A}}^*, s_{\mathrm{B}}^*\right) \in S_{\mathrm{A}} \times S_{\mathrm{B}} \text{ is a Pareto optimal iff (if and only if) } \nexists \text{ (there does not exist) another strategy profile } \mathrm{s} \in S_{\mathrm{A}} \times S_{\mathrm{B}} \text{ such that } \mathrm{u}(\mathrm{s}^*) \geq \mathrm{u}(\mathrm{s}) \text{ and at least } u_{\mathrm{A}}(\mathrm{s}^*) > u_{\mathrm{A}}(\mathrm{s}) \text{ or } u_{\mathrm{B}}(\mathrm{s}^*) > u_{\mathrm{B}}(\mathrm{s}). \tag{16.39}$$

It is obvious that s* = (withhold, withhold) satisfies Eq. 16.39.

If there were a unique Pareto optimal outcome for a cooperative game, that would seem to be a good solution concept. However, there is not. For example, in the prisoner's game, u (confess, withhold) = (–3, –10) is also a Pareto solution (see Example 16.8).

EXAMPLE 16.8

Verify that (confess, withhold) is a Pareto solution and (confess, confess) is not, using Eq. 16.39.

Solutions

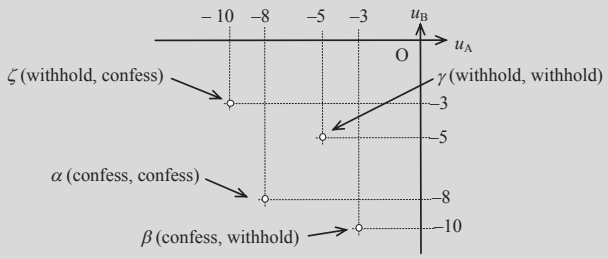
We first sketch the outcome or payoff in the so-called criterion space as shown below.

It is obvious that the prisoners are better off when their respective payoffs move closer to the origin O. As discussed in Section 16.5.2, the Nash equilibrium for a noncooperative game is point α, $u_\alpha = (-8, -8)$. Point γ is a Pareto solution as discussed. We are interested in finding out if point β is a Pareto solution. At point β (confess, withhold), $\mathrm{u}_\beta = (-3,-10)$. Now, we check if point β is a Pareto solution using Eq. 16.39.

We take point α, $\mathrm{u}_\alpha = (-8,-8)$. Is $\mathrm{u}_\beta = (-3,-10) \geq \mathrm{u}_\alpha(-8,-8)$? The answer is no. Next, we take point γ, $\mathrm{u}_\gamma = (-5,-5)$. Is $\mathrm{u}_\beta = (-3,-10) \geq \mathrm{u}_\gamma(-5,-5)$? The answer is no. Then, we take the final point ζ. $\mathrm{u}_\zeta = (-10,-3)$. Is $\mathrm{u}_\beta = (-3,-10) \geq \mathrm{u}_\zeta = (-10,-3)$? The answer is no. Therefore, there is no other point such that $\mathrm{u}_\beta \geq \mathrm{u}(s)$; from Eq. 16.39, $\mathrm{u}_\beta = (-3,-10)$ is Pareto optimal.

Now, we check if point α (confess, confess) is a Pareto solution using Eq. 16.39.

We take point β, $\mathrm{u}_\alpha = (-8,-8)$. Is $\mathrm{u}_\beta = (-8,-8) \geq \mathrm{u}_\alpha(-3,-10)$? The answer is no. Next, we take point ζ. $\mathrm{u}_\zeta = (-10,-3)$. Is $\mathrm{u}_\alpha = (-8,-8) \geq \mathrm{u}_\zeta = (-10,-3)$? The answer is no. Now, we take the final point γ, $\mathrm{u}_\gamma = (-5,-5)$. Is $\mathrm{u}_\alpha = (-8,-8) \geq \mathrm{u}_\gamma(-5,-5)$? The answer is yes. In fact, $\mathrm{u}_\alpha > \mathrm{u}_\gamma$. Therefore, according to Eq. 16.39, $\mathrm{u}_\alpha = (-8,-8)$ is not Pareto optimal.



In general, Pareto optimal or Pareto solutions are better understood in the so-called criterion space. In general, there could be infinitely many Pareto optima for any fairly complicated game. Pareto optimal and solution techniques will be formally introduced in Chapter 19 under the subject of multiobjective optimization. We use the Cournot duopoly model to illustrate Pareto optimal in Example 16.9.

EXAMPLE 16.9

Continue with Example 16.5, except formulating the problem as a cooperative game and looking for the Pareto optimal. From Example 16.5 we have the payoff functions for firms 1 and 2 as, respectively,

$$u_1(q_1, q_2) = -(q_1)^2 + (A - q_2 - c_1)q_1, \text{ and} \tag{16.40}$$

$$u_2(q_1, q_2) = -(q_2)^2 + (A - q_1 - c_2)q_2 \tag{16.41}$$

We assume $A = 200$, $c_1 = c_2 = 2$. Hence, Eqs 16.40 and 16.41 become

$$u_1(q_1, q_2) = -(q_1)^2 + (198 - q_2)q_1, \text{ and} \tag{16.42}$$

$$u_2(q_1, q_2) = -(q_2)^2 + (198 - q_1)q_2 \tag{16.43}$$

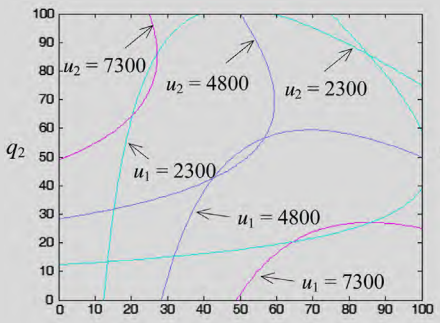
Solutions

The profit functions u_1 and u_2 are graphed in terms of q_1 and q_2 as shown on the next page (left), indicating that the maxima of u_1 and u_2 are approaching in the opposite direction; u_1 toward the lower right and u_2 toward the upper left. In this case, $q_1 - q_2$ is called the design space. Finding the Pareto optimal in a design space is not straightforward and is not desired. Instead, we graph u_1 and u_2 in the so-called criterion space shown on the next page (right). MATLAB scripts for creating these graphs can be found in Appendix A (at the book's companion site).

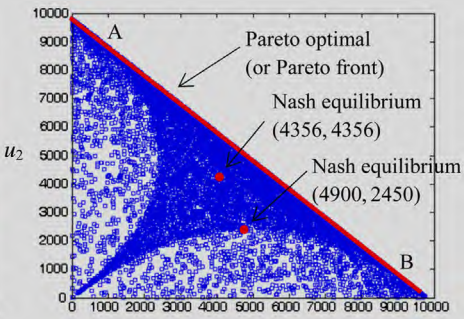
Based on the definition of Eq. 16.39, points along the line between A and B are Pareto optimal, also called the Pareto front.

In the noncooperative game shown in Example 16.5, the Nash equilibrium is found at $q_1^* = q_2^* = 66$ and $u_1 = u_2 = 4356$, as shown in the figure below (right), which is not Pareto optimal. Any point on the front is Pareto optimal, in which neither firm is better off without making the other firm worse off.

Also from Example 16.7, the Nash equilibrium of a sequential game is found at $q_1^* = 99$ and $q_2^* = 49.5$ profits $u_1 = 4900.5$, and $u_2 = 2.450.25$, which is not Pareto optimal either.



Design space



Criterion space

16.6 DESIGN EXAMPLES

We discussed two major theories—utility theory and game theory—which were explored to support engineering design. In this section, we use two engineering design examples to illustrate the details. We first discuss using utility theory as a design tool and apply the design tool to solve a simple cantilever beam example. Then, we use game theory as a design tool to solve a pressure vessel design problem, which is formulated as noncooperative, sequential, and cooperative games for illustration. Both examples are purposely made simple so that the focus stays on concept illustration.

16.6.1 UTILITY THEORY AS A DESIGN TOOL

In this subsection, we illustrate a few details in employing utility theory to support multiobjective engineering designs. We use a simple cantilever beam example of two design attributes (or design objectives) to solve for two design problems, unconstrained and constrained.

A cantilever beam of rectangular cross section is loaded by a point force $P = 1000$ N at the tip, as shown in Figure 16.14. The length of the beam is $\ell = 200$ mm, the width is $b = 30$ mm, and height is $h = 40$ mm. The beam is made of aluminum 1060 Alloy, in which the yield strength is $S_y = 27.6$ MPa, modulus of elasticity is $E = 69$ GPa $= 69{,}000$ MPa, and mass density is $\rho = 2.7 \times 10^{-3}$ g/mm^3.

The weight w, bending stress σ, and vertical displacement z are included for design consideration. They are defined respectively as follows:

$$w = \rho g b h \ell \tag{16.44}$$

where $g = 9806$ mm/s^2 is the gravitational acceleration,

$$\sigma = \frac{6P\ell}{bh^2} \tag{16.45}$$

and

$$z = \frac{4P\ell^3}{Ebh^3} \tag{16.46}$$

For the current design, $b = 30$ mm and $h = 40$ mm; the weight, the maximum bending stress, and maximum vertical displacement are, respectively, $w = 6.35$ N, $\sigma = 25$ MPa, and $z = 0.2415$ mm. Note that weight is very small compared to the external load P; therefore, self-weight is ignored in stress and

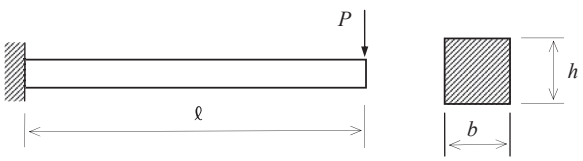


FIGURE 16.14

Cantilever beam example.

displacement calculations. Also, the maximum bending stress is less than the material yield strength S_y. Next, we introduce two design problems, unconstrained and constrained, and illustrate the concept of solving the design problems using utility theory.

16.6.1.1 Beam Design Example 1: Unconstrained Problem

The two objective functions included in this design problem are weight $w(b,h)$ and displacement $z(b,h)$, in which the width b and height h of the beam cross-section are defined as design variables. Note that the length of the beam is assumed to be fixed. The lower and upper bounds of the design variables are chosen as $10 \text{ mm} \leq b \leq 60 \text{ mm}$, and $20 \text{ mm} \leq h \leq 80 \text{ mm}$.

We illustrate the solution in four steps and present results for three cases.

Step 1: Determination of attribute bounds

We determine the bounds—that is, the worst and best designs—of the attributes (in this example, the two objectives) based on the bounds of design variables b and h. For the weight objective, the worst (heaviest) and best designs are, respectively:

$w_{\text{worst}} = 25.4$ N, when both design variables reach their respective upper bounds; that is, $b = 60$ mm and $h = 80$ mm; and
$w_{\text{best}} = 1.06$ N, when both design variables reach their respective lower bounds; that is, $b = 10$ mm and $h = 20$ mm.

For the displacement objective, the worst (largest displacement) and best designs are, respectively:

$z_{\text{worst}} = 5.80$ mm, when both design variables reach their respective lower bounds; that is, $b = 10$ mm and $h = 20$ mm; and
$z_{\text{best}} = 0.0151$ mm, when both design variables reach their respective upper bounds; that is, $b = 60$ mm and $h = 80$ mm.

It is obvious that these two objectives are in conflict.

Step 2: Formulation of SAU functions

We define two SAU functions for the respective two attributes following the utility functions defined in Section 16.2.4. For the weight attribute, following Eq. 16.14, we have

$$u_w(s) = \frac{1 - e^{-r_w s_w}}{1 - e^{-r_w}} \tag{16.47}$$

where s_w is a normalized form of the outcome in the weight attribute, as defined in Eq. 16.15; that is,

$$s_w = \frac{w - w_{\text{worse}}}{w_{\text{best}} - w_{\text{worse}}} = \frac{w - 25.4}{1.06 - 25.4} \tag{16.48}$$

For the displacement attribute, we have

$$u_z(s) = \frac{1 - e^{-r_z s_z}}{1 - e^{-r_z}} \tag{16.49}$$

where s_z is a normalized form of the outcome in the displacement attribute, defined as

$$s_z = \frac{z - z_{\text{worse}}}{z_{\text{best}} - z_{\text{worse}}} = \frac{z - 5.80}{0.0151 - 5.80} \tag{16.50}$$

Note that, as discussed in Section 16.4.4, the parameters r_w and r_z determine the shape of the respective utility function curves, which reflect the designer's attitude toward risk. Recall that r_w (or r_z) > 0 implies risk averse, r_w (or r_z) < 0 implies risk prone, and r_w (or r_z) $= 0$ implies risk neutral. The utility curves of corresponding situations for the two respective attributes are shown in Figure 16.15.

Step 3: Formulation of MAU functions

We define a multiplicative MAU by introducing scaling constants k_w and k_z that represent the designer's preference between attributes w and z. The multiplicative MAU is defined, following Eq. 16.17, as

$$\begin{aligned} u(w,z) &= \frac{1}{K}\left\{\prod_{i=1}^{2}[1 + Kk_i u_i] - 1\right\} = \frac{1}{K}\{(1 + Kk_w u_w)(1 + Kk_z u_z) - 1\} \\ &= k_w u_w + k_z u_z + Kk_w k_z u_w u_z \end{aligned} \tag{16.51}$$

in which K satisfies

$$1 + K = \prod_{i=1}^{2}(1 + Kk_i) = (1 + Kk_w)(1 + Kk_z) \tag{16.52}$$

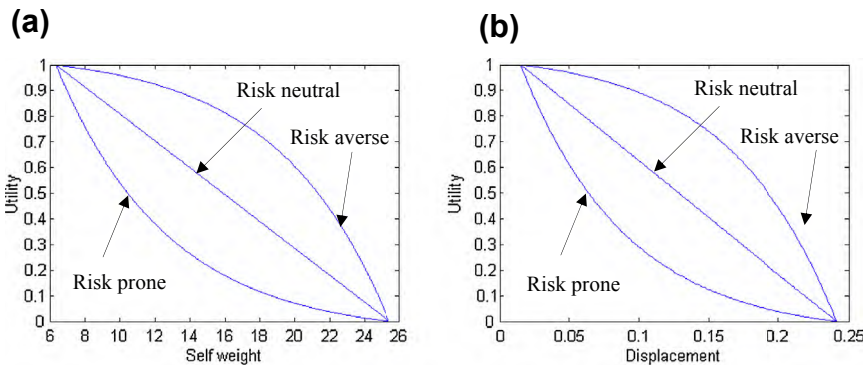


FIGURE 16.15

Utility functions: (a) for attribute: weight w, and (b) for attribute: displacement z.

That is,

$$K = \frac{1 - k_w - k_z}{k_w k_z} \tag{16.53}$$

Our goal is now to maximize the MAU function $u(w,z)$.

Step 4: Search Designs

In this step, we introduce three cases illustrating different aspects of the design scenarios that present concept and essential elements of applying utility theory to engineering design.

Case 1: Base case—Risk neutral ($r_w = r_z = 0$) and no preference ($k_w = 0.5$, $k_z = 0.5$).

As discussed in Section 16.4.4, for $r_w = r_z = 0$, the SAU functions are respectively $u_w = s_w$ and $u_z = s_z$. And for $k_w = 0.5$, $k_z = 0.5$, the multiplicative MAU becomes additive MAU; that is, $u(w,z) = k_w u_w + k_z u_z = 0.5\ (s_w + s_z)$:

$$u(w,z) = k_w u_w + k_z u_z = 0.5(s_w + s_z) = 0.5\left(\frac{w - 25.4}{1.06 - 25.4} + \frac{z - 5.80}{0.0151 - 5.80}\right) \tag{16.54}$$

in which w and z are the weight and displacement of the beam defined in Eqs 16.44 and 16.46, respectively. Note that both w and z are functions of design variables b and h.

There are numerous methods to find the maximum of the MAU function defined in Eq. 16.54. Finding maximum or minimum of the objective function in a design problem is called design optimization, which will be discussed in Chapter 17. For this example, we use a brute force approach, the so-called generative method, to find the maximum of the MAU function. In this method, the design variables are divided into number of intervals between their respective upper and lower bounds, from which the maximum is sought by comparing the values of the MAU function at each design to the combination of the design variables in the intervals.

In this example, we define a design interval as $d = 0.1$ mm for both b and h. As a result, design variables b and h are divided into, respectively, $n_b = (b_u - b_\ell)/d + 1 = (60 - 10)/0.1 + 1 = 501$, and $n_h = (h_u - h_\ell)/d + 1 = (80 - 20)/0.1 + 1 = 601$. The MAU function will be evaluated $nb \times nh = 501 \times 601$ times. We use MATLAB to find the maximum and create graphs to illustrate the results. The optimum design is found at $b = 10$ mm and $h = 57.7$ mm, in which the MAU function is $u(w,z) = 0.9395$, and the weight and displacement of the beam are, respectively, $w = 3.06$ N and $z = 0.241$ mm. The solution graphs are shown in Figure 16.16, and the MATLAB script is given in Appendix A (at the book's companion site).

Case 2: Changing preference between attributes w and z.

In this case, we assume risk neutral, but change the preference attributes of w and z. In Case 2a, we assume $k_w = 0.7$, $k_z = 0.3$, and for Case 2b, we assume $k_w = 0.3$, $k_z = 0.7$.

The optimum design for Case 2a is found at $b = 10$ mm and $h = 46.7$ mm, in which the MAU function is $u(w,z) = 0.9365$, and the weight and displacement of the beam are, respectively, $w = 2.47$ N and $z = 0.455$ mm. The solution graphs are shown in Figure 16.17(a).

The optimum design for Case 2b is found at $b = 10$ mm and $h = 71.3$ mm, in which the MAU function is $u(w,z) = 0.9529$, and the weight and displacement of the beam are, respectively, $w = 3.76$ N and $z = 0.128$ mm. The solution graphs are shown in Figure 16.17(b).

Comparing Cases 2a and 2b to Case 1, it is clear that a larger preference factor pushes the solution toward a design that reflects the designer's preference, as illustrated in Figure 16.18. For example, for

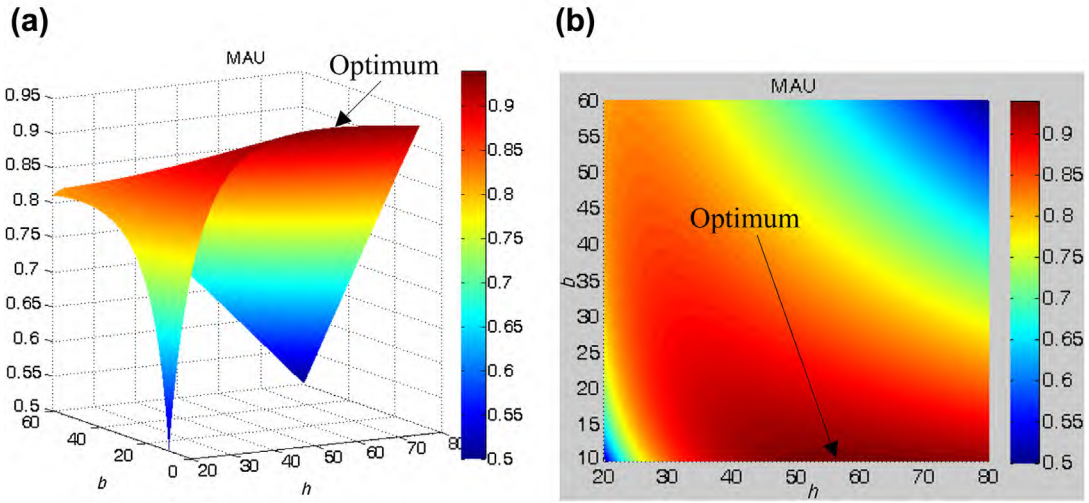


FIGURE 16.16

MAU function and optimum solution of Case 1: (a) iso-view and (b) top view.

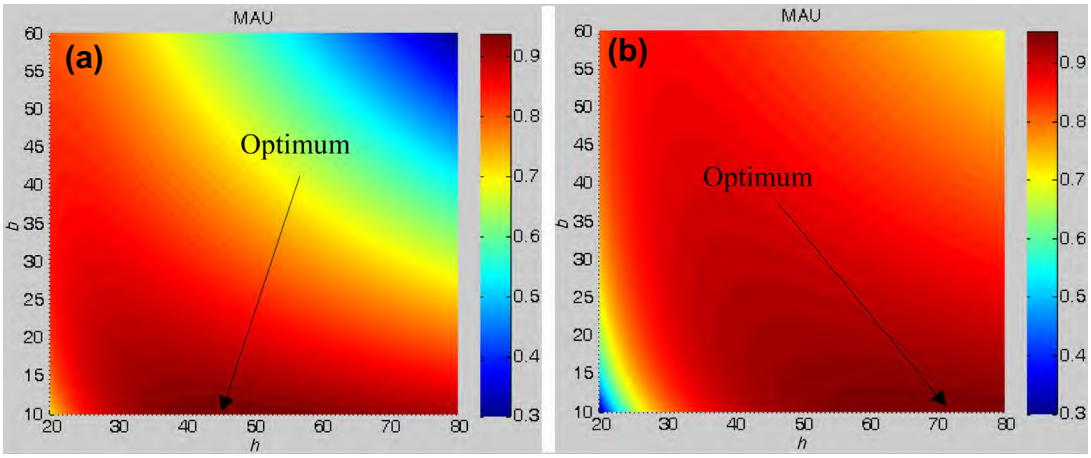


FIGURE 16.17

MAU function and optimum solution of Case 2: (a) Case 2a, $k_w = 0.7$, $k_z = 0.3$. (b) Case 2b, $k_w = 0.3$, $k_z = 0.7$.

Case 2a, preference is given to weight attribute ($k_w = 0.7$); the optimal design in weight becomes less compared to Case 1. On the other hand, for Case 2b, preference is given to displacement attribute ($k_z = 0.7$); the optimal design in displacement becomes smaller compared to Case 1.

Case 3: Changing attitude toward risk for weight *w*.

In this case, we assume no preference between attributes *w* and *z* (i.e., $k_w = k_z = 0.5$), but change the attitude toward risk for the weight attribute to both risk prone ($r_w = -2$, Case 3a), and risk averse ($r_w = 2$, Case 3b).

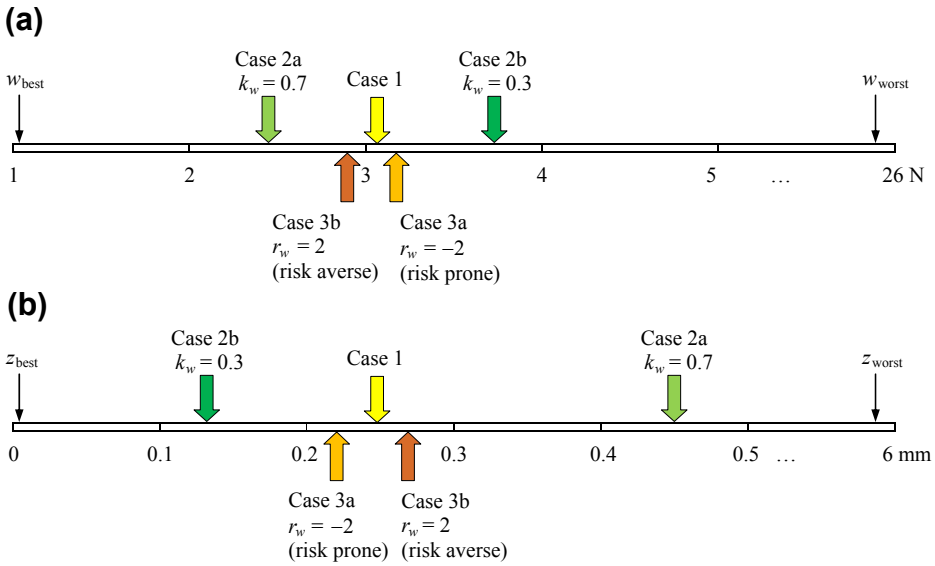


FIGURE 16.18

Comparison of MAU function and optimum solutions of Cases 1, 2, and 3: (a) weight attribute and (b) displacement attribute.

The optimum design for Case 3a is found at $b = 10$ mm and $h = 59.1$ mm, in which the MAU function is $u(w,z) = 0.9394$, and the weight and displacement of the beam are, respectively, $w = 3.13$ N and $z = 0.225$ mm. The solution graphs are shown in Figure 16.19(a).

The optimum design for Case 3b is found at $b = 10$ mm and $h = 56.6$ mm, in which the MAU function is $u(w,z) = 0.9394$, and the weight and displacement of the beam are respectively, $w = 2.99$ N and $z = 0.256$ mm. The solution graphs are shown in Figure 16.19(b).

Comparing Cases 3a and 3b to Case 1, it is clear that a risk-prone (Case 3a) designer yields a design with larger weight, and a risk-averse (Case 3b) designer yields a design with smaller weight, as expected.

Results of the five cases (1, 2a, 2b, 3a, and 3b) are summarized in Table 16.12. In all five cases, at the optimum, the beam width is at its minimum; only the height h is different. Increasing the height h increases weight attribute but decreases the displacement attribute z. Preference or risk attitude that favors the weight attribute, including Cases 2a and 3b, leads to a smaller height, and hence less weight. On the other hand, preference or risk attitude that favors the displacement attribute, including Cases 2b and 3a, leads to a larger height, hence a smaller displacement.

16.6.1.2 Beam Design Example 2: Constrained Problem

This example is identical to that of beam design example 1, except that we add a stress constraint; that is, $\sigma < S_y = 27.6$ MPa. One of the approaches in handling the constrained design problem in the framework of utility theory is to first construct an SAU function for the constraint function, and

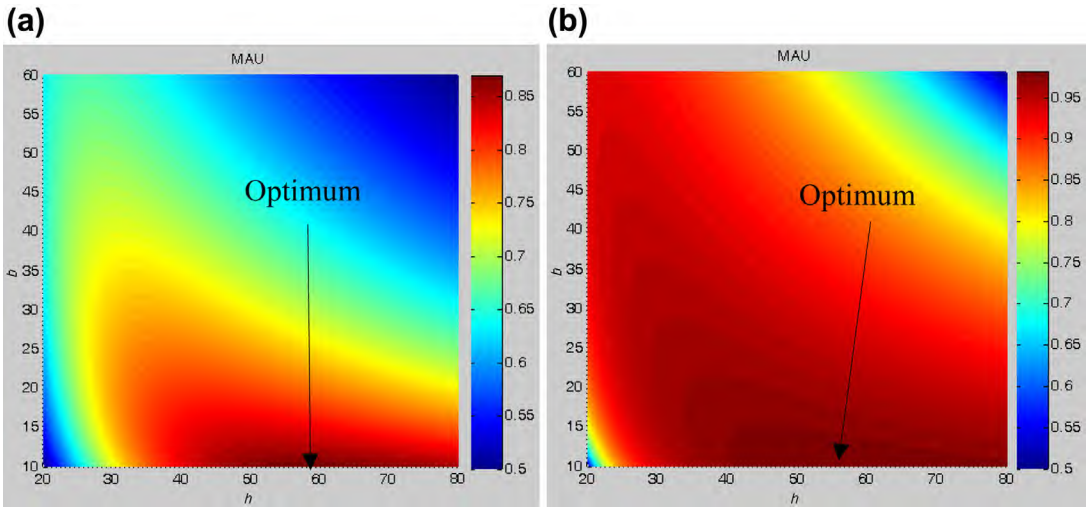


FIGURE 16.19

MAU function and optimum solution of Case 3: (a) Case 3a, $r_w = -2$ (risk prone) and (b) Case 3b (risk averse), $r_w = 2$.

incorporate the SAU to the MAU function, in which the infeasible design (where constraint is violated) is automatically eliminated from design consideration.

We follow the same four steps as those of the previous example to set up and solve the constrained problem.

Step 1: Determination of attribute bounds

The bounds for the weight and displacement remain unchanged. For the stress constraint, the worst largest) and best designs are, respectively:

$\sigma_{\text{worst}} = 300$ MPa, when both design variables reach their respective lower bounds; that is, $b = 10$ mm and $h = 20$ mm; and
$\sigma_{\text{best}} = 3.13$ MPa, when both design variables reach their respective upper bounds; that is, $b = 60$ mm and $h = 80$ mm.

Table 16.12 Result Comparison for Unconstrained Design Problems

	Problem Setup					Results			
Case No.	k_w	k_z	r_w	r_z	b (mm)	h (mm)	w (N)	z (mm)	u
Case 1	0.5	0.5	0	0	10	57.5	3.06	0.241	0.939
Case 2a	0.7	0.3	0	0	10	46.7	2.47	0.455	0.937
Case 2b	0.3	0.7	0	0	10	71.3	3.78	0.128	0.953
Case 3a	0.5	0.5	−2	0	10	59.1	3.13	0.225	0.939
Case 3b	0.5	0.5	2	0	10	56.6	2.99	0.256	0.939

Step 2: Formulation of SAU functions

Again, we only need to construct a utility function for the stress constraint. The utility function must reveal the nature of the constraint; that is, utility is 1 when stress is less than the yield strength (also called feasible design) and 0 when the stress is over the upper limit (infeasible design). One function that satisfies the requirement is a unit step function. However, a step function is discontinuous and is difficult to handle. Therefore, we introduce a function that is continuous and almost reveals the nature of a step function as follows:

$$u(\sigma) = \frac{1}{1 + e^{\frac{\sigma_{0.5} - \sigma}{s}}} \tag{16.55}$$

in which $\sigma_{0.5}$ is the value of the stress at which $u(\sigma) = 0.5$, and s is the slope of the utility function $u(\sigma)$ at utility $u = 0.5$. If we set $\sigma_{0.5} = S_y = 27.6$ MPa, the utility functions for slope $s = -1$ and $s = -0.1$ are graphed in Figures 16.20(a) and (b), respectively. It is clear that when the slope is small, the utility function closely resembles a unit step function as desired. In this example, we set slope $s = -0.01$.

Step 3: Formulation of MAU functions

We define a multiplicative MAU by adding the utility function of the stress constraint and introducing an additional parameter k_c as follows (Iyer et al., 1999):

$$u(w, z, \sigma) = [k_c + (1 - k_c)u(w, z)]u(\sigma) \tag{16.56}$$

in which the utility function $u(w,z)$ is defined in Eq. 16.51. Eq. 16.56 shows that the utility function $u(w,z,\sigma) = 0$ when $u(\sigma) = 0$, implying that an infeasible design is encountered. For a feasible design, because $u(\sigma) = 1$ as defined in Eq. 16.55, the MAU function $u(w,z,s) = k_c + (1 - k_c)u(w,z)$ with $k_c < 1$. Note that k_c represents the utility of a feasible design with worst possible attribute values, which is introduced to distinguish the situations between a feasible design of zero utility and an infeasible design where utility is always zero, as defined in Eq. 16.56. In this example, we set $k_c = 0.2$.

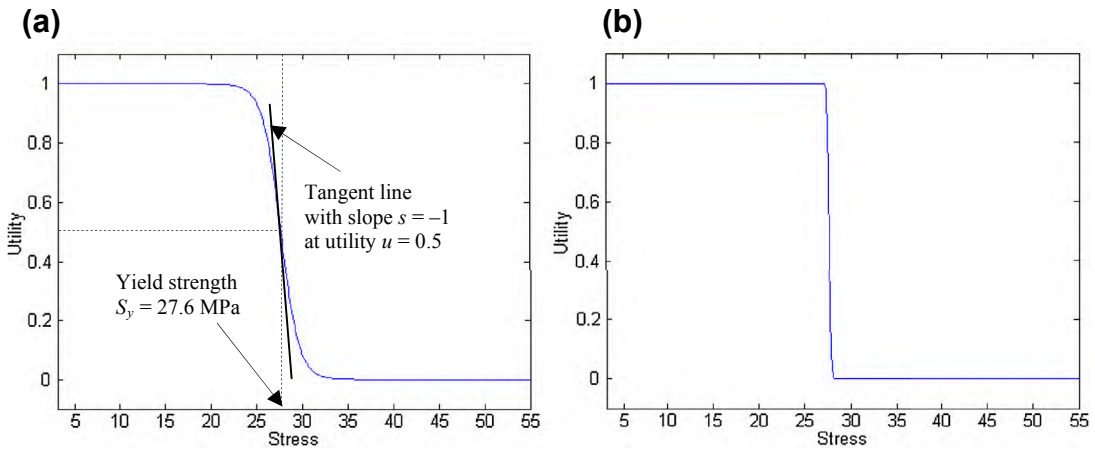


FIGURE 16.20

SAU function for a stress constraint: (a) slope $s = -1$ and (b) slope $s = -0.1$.

Step 4: Search Designs

In this step, we introduce three cases illustrating different aspects of the design scenarios.

Case 4: Base case—Risk neutral ($r_w = r_z = 0$) and no preference ($k_w = 0.5$, $k_z = 0.5$)

The optimum design is found at $b = 10$ mm and h =66.1 mm, in which the MAU function is $u(w,z) = 0.9498$, and the weight and displacement of the beam are, respectively, $w = 3.50$ N and $z = 0.161$ mm. The solution graphs are shown in Figure 16.21, and the MATLAB script is given in Appendix A (at the book's companion site).

Recall in Case 1 that the optimum is found for the unconstrained problem at $b = 10$ mm and $h = 57.7$ mm. Obviously, for the constrained problem, when the stress constraint is taken into consideration, the height of the beam cross-section is increased to $h = 66.1$ mm in order to bring the design into the feasible range.

Cases 2 and 3 are repeated with the stress constraint as Cases 5 and 6. That is, in Case 5a, we assume $k_w = 0.7$, $k_z = 0.3$; for Case 5b, we assume $k_w = 0.3$, $k_z = 0.7$. In Case 6a, we change the attitude toward risk for the weight attribute to risk prone ($r_w = -2$); and in Case 6b, to risk averse ($r_w = 2$). Results are listed in Table 16.13. Results of Cases 5a, 6a, and 6b are identical to those of Case 4. This is because the weight cannot be reduced further without violating the stress constraint, although a preference is given to the weight attribute (Case 5a) and risk prone toward weight attribute is

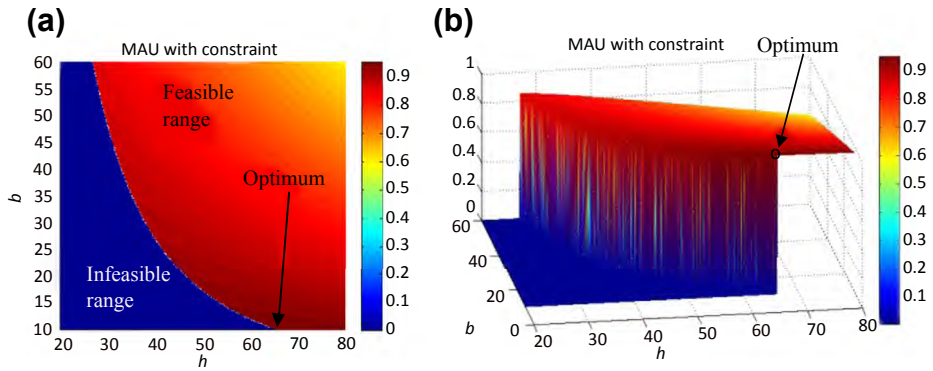


FIGURE 16.21

MAU function and optimum solution of Case 4: (a) top view and (b) iso-view.

Table 16.13 Result Comparison for the Constrained Design Problems

Case No.	Problem Setup				Results					
	k_w	k_z	r_w	r_z	b (mm)	h (mm)	w (N)	z (mm)	σ (MPa)	u
Case 4	0.5	0.5	0	0	10	66.1	3.50	0.161	27.46	0.950
Case 5a	0.7	0.3	0	0	10	66.1	3.50	0.161	27.46	0.938
Case 5b	0.3	0.7	0	0	10	71.3	3.78	0.128	23.60	0.962
Case 6a	0.5	0.5	−2	0	10	66.1	3.50	0.161	27.46	0.906
Case 6b	0.5	0.5	2	0	10	66.1	3.50	0.161	27.46	0.977

assumed (Case 6a). On the other hand, displacement can be further reduced by increasing the height h without violating the stress constraint. As a result, we observe displacement reduction due to the height increment in Case 5b.

Comparing Cases 1 and 4, with and without the stress constraint, the height h cannot be reduced to less than 66.1 mm without violating the stress constraint, as shown in the results of Case 4. The same observation is found in the remaining cases, except for Cases 2 and 5b, in which the same results are found: displacement is reduced by increasing height h without violating the stress constraint.

16.6.1.3 Summary of Utility Theory as a Design Tool

Employing utility theory to support engineering design offers several advantages. First, SAU utility functions that normalize the respective attributes are constructed so that these attributes can be treated with uniformity. Second, a designer's attitude toward risk is captured and incorporated into design decision making. Third, a designer's preferences are well represented in a MAU function, either in an additive or multiplicative form. Also, constraints can be handled by constructing a continuous function closely resembling a step function, which brings the utility of the constraint function into the MAU function.

Although utility theory offers a plausible approach for support of engineering design, selecting proper utility function and determining parameters that reflect a designer's preferences and attitude toward risk remain uncertain. From the perspective of multiobjective optimization (to be discussed in Chapter 19), by maximizing one single MAU function that is converted by combining individual SAU functions represents one of the solution techniques, referred to as methods with a priori articulation of preferences. The pros and cons of methods with a priori articulation of preferences for multiobjective optimization will be discussed in Chapter 19, along with methods other than those with a priori preferences. The beam examples will be revisited in the context of multiobjective optimization in Chapter 19.

16.6.2 GAME THEORY AS A DESIGN TOOL

A cylindrical pressure vessel example is discussed in this subsection to illustrate the idea of using game theory as a design tool. We formulate the design problem as three different games: a strategy form game, a sequential game, and a cooperative game. To minimize computation and mathematical equations involved, we use graphs in design space and criterion space to facilitate the concept illustration.

16.6.2.1 The Pressure Vessel Design Example

A pressure vessel shown in Figure 16.22 is internally pressurized with pressure $P = 4000$ psi. The length, radius, and thickness of the vessel are, respectively, $\ell = 100$ in., $R = 30$ in., and $t = 1$ in. The weight density and yield strength of the vessel material are, respectively, $\gamma = 0.283$ lb/in. and $S_y = 32{,}000$ psi.

The design objectives are to minimize the weight W and maximize the volume V of the vessel by varying two design variables, radius R and thickness t; that is,

$$\text{Minimize:} \quad W(R,t) = \gamma\left[\pi(R+t)^2(\ell+2t) - \pi R^2\ell\right] \tag{16.57}$$

$$\text{Maximize:} \quad V(R,t) = \pi R^2\ell \tag{16.58}$$

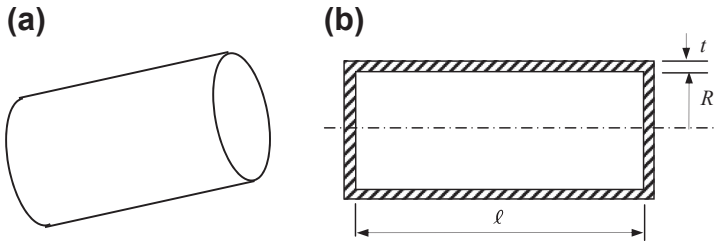


FIGURE 16.22

Cylindrical pressure vessel: (a) iso-view and (b) section view with dimensions.

The constraints include the hoop stress and a number of inequality constraints on the geometric dimensions of the design variables, as listed below:

$$\sigma(R,t) = PR/t \leq S_y \tag{16.59}$$

$$5t - R \leq 0 \tag{16.60}$$

$$t + R \leq 40 \tag{16.61}$$

In addition, sides constraints specify the upper and lower bounds for the respective design variables R and t; that is,

$$0.5 \leq t \leq 6 \text{ in.} \tag{16.62}$$

$$0 \leq R \leq 38 \text{ in.} \tag{16.63}$$

The constraints defined in Eqs 16.59 and 16.60 can be combined and simplified into the following constraint:

$$5t \leq R \leq 8t \tag{16.64}$$

Therefore, the feasible region can be identified as a polygon ABCDE by intersecting the four constraint equations (Eqs 16.61 to 16.64), and it is graphed in Figure 16.23. Note that the intersecting points are calculated as A(0.5, 2.5), B(0.5, 4), C(4.44, 35.55), D(6, 34), and E(6, 30).

Note that in Figure 16.23, the upper and lower bounds of the design variables R and t are represented by dashed lines. Apparently, due to constraints other than the side constraints, the original upper and lower bounds of R will never be activated. The feasible bounds for R are now reduced to

$$2.5 \leq R \leq 35.55 \text{ in.} \tag{16.65}$$

which will be automatically satisfied when all other constraints are not violated. Therefore, the design problem is reduced to Eqs 16.57, 16.58, 16.61, 16.62, 16.64, and 16.65.

We assume two players (or designers) are playing (or providing design decisions) this game (or design problem). Player W wishes to minimize the weight of the vessel W by varying the thickness design variable t. Player V wishes to maximize the volume of the vessel V by varying the radius design variable R.

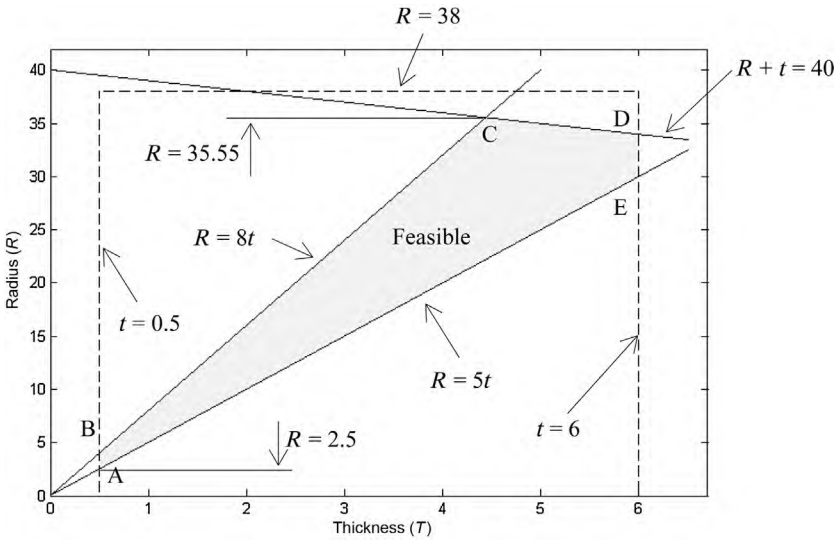


FIGURE 16.23

The feasible range of the pressure vessel design problem.

16.6.2.2 Strategy Form Game

In this game, we assume that players are noncooperative. They are varying their respective design variable for the best possible objective that can be achieved. The solution to this problem is a Nash equilibrium or Nash solution, as discussed in Section 16.5.2.

First, player V needs to maximize V by changing R for any given strategy of player W; that is,

$$\text{Minimize: } -V = -\pi R^2 \ell \tag{16.66}$$

$$\text{Subject to: } t + R \leq 40 \tag{16.67}$$

$$5t \leq R \leq 8t \tag{16.68}$$

$$2.5 \leq R \leq 35.55 \tag{16.69}$$

Note that $-V$ decreases monotonically with increasing R, which means player V needs to choose the largest possible R. Therefore, the solution (also called the rational reaction set in literature) is formulated as below, by reviewing constraint equations (Eqs 16.67 and 16.68):

$$R^*(t) = \begin{cases} 40 - t, & \text{if } 4.45 \leq t \leq 6 \\ 8t, & \text{if } 0.5 \leq t \leq 4.45 \end{cases} \tag{16.70}$$

or

$$R^*(t) = \max\{40 - t, 8t\} \tag{16.71}$$

which depends on the thickness design variable t, determined by player W.

Note that this solution can be easily obtained using the feasible region diagram (red line BCD in Figure 16.24(a)), requiring player V to maximize R for each t within the feasible range.

In the meantime, player W needs to minimize weight of the vessel W (by changing t) for any given strategy of player V; that is

$$\text{Minimize: } W = \rho\left[\pi(R+t)^2(\ell+2t) - \pi R^2\ell\right] \tag{16.72}$$

$$\text{Subject to: } t + R \leq 40 \tag{16.73}$$

$$5t \leq R \leq 8t \tag{16.74}$$

$$0.5 \leq t \leq 6 \text{ in.} \tag{16.75}$$

Note that W decreases monotonically with decreasing t, which means player W needs to choose the smallest possible t. Therefore, the solution (rational reaction set) can be obtained as

$$t^*(R) = \begin{cases} R/8, & \text{if } 4 \leq R \leq 35.5 \\ 0.5, & \text{if } 2.5 \leq R \leq 4 \end{cases} \tag{16.76}$$

or

$$t^*(R) = \min\{0.5,\ R/8\} \tag{16.77}$$

which depends on R.

Again, the solution can be easily observed from the feasible space diagram (blue line ABC in Figure 16.24(b)); that is, player W needs to minimize t for each R within the feasible range.

The Nash solution is where the rational reaction sets of the two players intersect with each other. In this case, the Nash solution is $R = 8t$, and $4 \leq R \leq 35.5$, which is represented by the straight line segment BC (green) in the design space, as shown in Figure 16.25.

16.6.2.3 Sequential Game

We present two cases: Case 1, in which player W is the leader, and Case 2, in which player V is the leader.

Case 1: Player W is the leader.

As discussed in the previous section, the design problem for player W is

$$\text{Minimize: } W = \rho\left[\pi(R+t)^2(\ell+2t) - \pi R^2\ell\right] \tag{16.78}$$

$$\text{Subject to: } R = R^*$$

$$0.5 \leq t \leq 6 \text{ in.} \tag{16.79}$$

where R^* is the solution to the follower's (player V) problem, given by

$$\text{Minimize: } -V = -\pi R^2\ell \tag{16.80}$$

$$\text{Subject to: } t + R \leq 40 \tag{16.81}$$

$$5t \leq R \leq 8t \tag{16.82}$$

$$2.5 \leq R \leq 35.55 \tag{16.83}$$

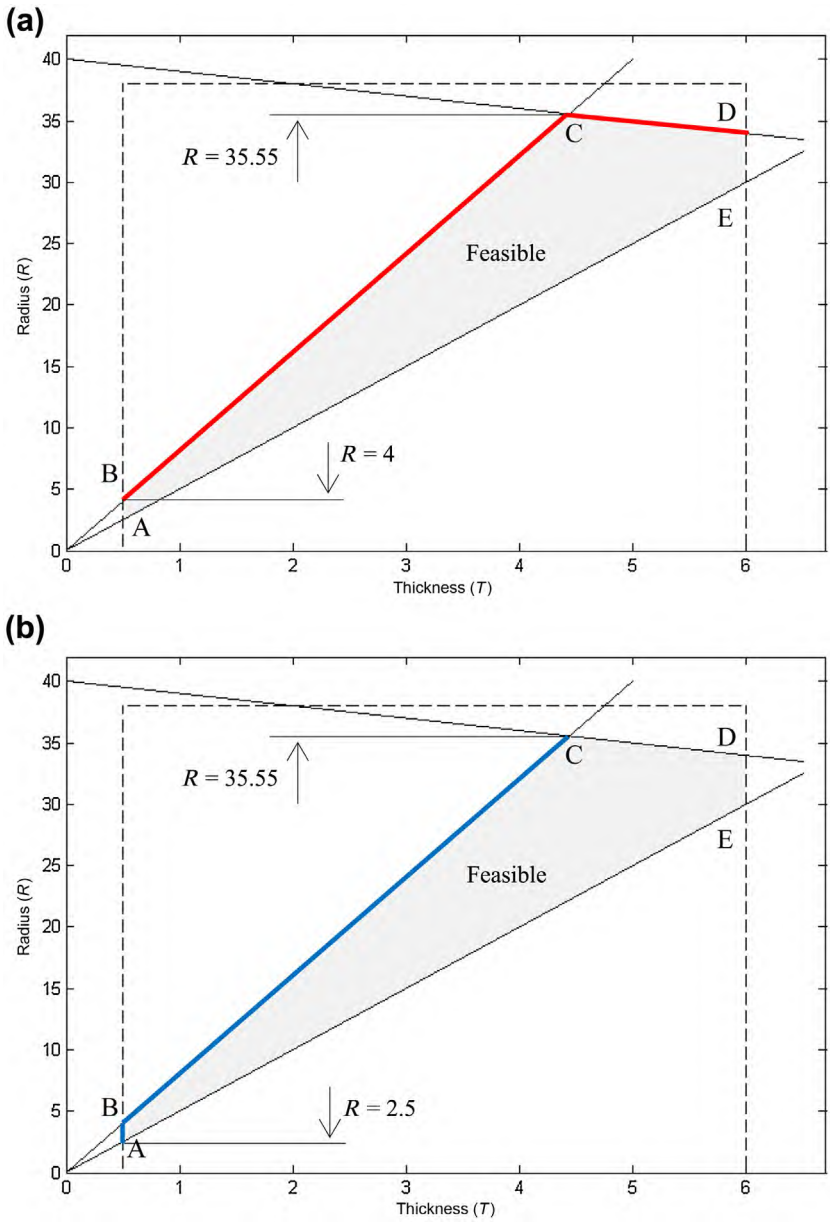


FIGURE 16.24

Exploration of design solutions in the design space: (a) solutions for player *W* and (b) solutions for player *V*.

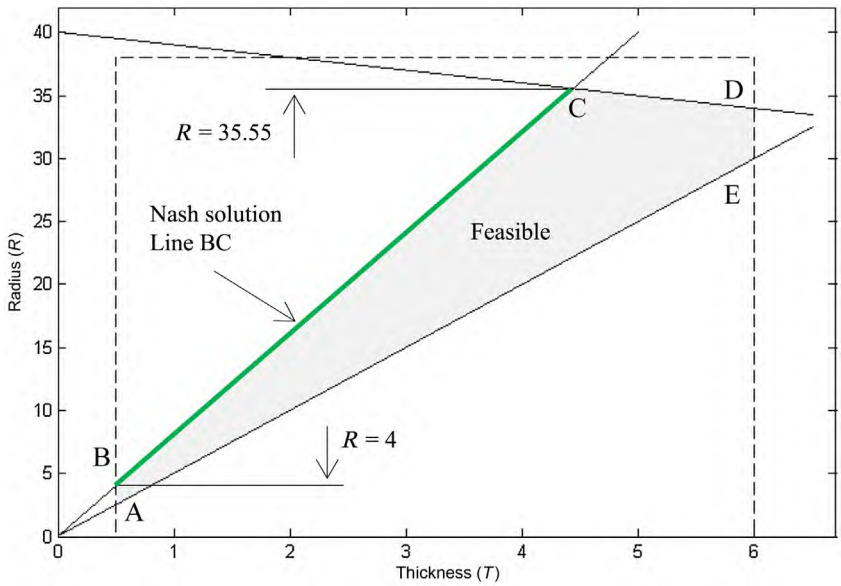


FIGURE 16.25

Nash solution represented on line segment BC.

Note that the follower's problem has been solved above when calculating the Nash solution, as

$$R^*(t) = \max\{40 - t, 8t\} \tag{16.84}$$

Substituting R^* into the leader's design problem, the leader's objective function W then becomes a function of t only, which is plotted in Figure 16.26(a).

Because $W(t)$ increases monotonically with respect to t, the solution to the leader's problem (minimize W) is $t = 0.5$ in.; hence, $R = 8t = 4$ in. This solution represents a very long vessel with small thickness and radius.

Case 2: Player V is the leader.

As discussed in the previous section, the design problem for player V is

$$\text{Minimize: } -V = -\pi R^2 \ell \tag{16.85}$$

$$\text{Subject to: } t = t^*$$

$$2.5 \leq R \leq 35.55 \tag{16.86}$$

where t^* is the solution to the follower's (player W) problem, given by

$$\text{Minimize: } W = \rho\left[\pi(R+t)^2(\ell + 2t) - \pi R^2 \ell\right] \tag{16.87}$$

$$\text{Subject to: } t + R \leq 40 \tag{16.88}$$

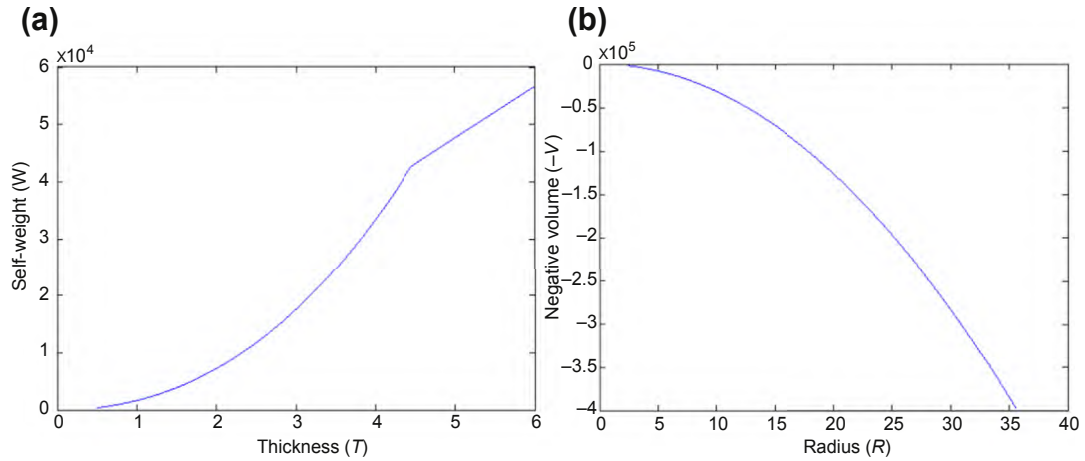


FIGURE 16.26

Illustration of solutions of the sequential game: (a) player W is the leader and (b) player V is the leader.

$$0.5 \leq t \leq 6 \text{ in.} \tag{16.89}$$

$$5t \leq R \leq 8t \tag{16.90}$$

Note that the follower's problem has been solved above when calculating the Nash solution, as $t^*(R) = \min\{0.5, R/8\}$.

Substituting t^* into the leader's design problem, the leader's objective function $-V$ then becomes a function of R only, which is plotted in Figure 16.26(b).

Because $-V(R)$ decreases monotonically with respect to R, the solution to the leader's problem (minimize $-V$) is $R = 35.5$ in.; hence, $t = R/8 = 4.44$ in. This solution represents a narrow vessel with large thickness and radius.

Neither a very long vessel with small thickness and radius obtained from the game that player W is the leader nor a narrow vessel with large thickness and radius (player V is the leader) is desirable in the design of the pressure vessel. A compromised design is generally considered to be better than these two extremes.

16.6.2.4 Cooperative Game

As discussed in Section 16.5.4, the solution to a cooperative game is Pareto optimal. The design problem is restated as below.

$$\text{Minimize: } W(R,t) = \gamma \left[\pi(R+t)^2(\ell+2t) - \pi R^2 \ell\right] \tag{16.91}$$

$$\text{Maximize: } V(R,t) = \pi R^2 \ell \tag{16.92}$$

$$\text{Subject to: } t + R \leq 40 \tag{16.93}$$

$$0.5 \leq t \leq 6 \text{ in.} \tag{16.94}$$

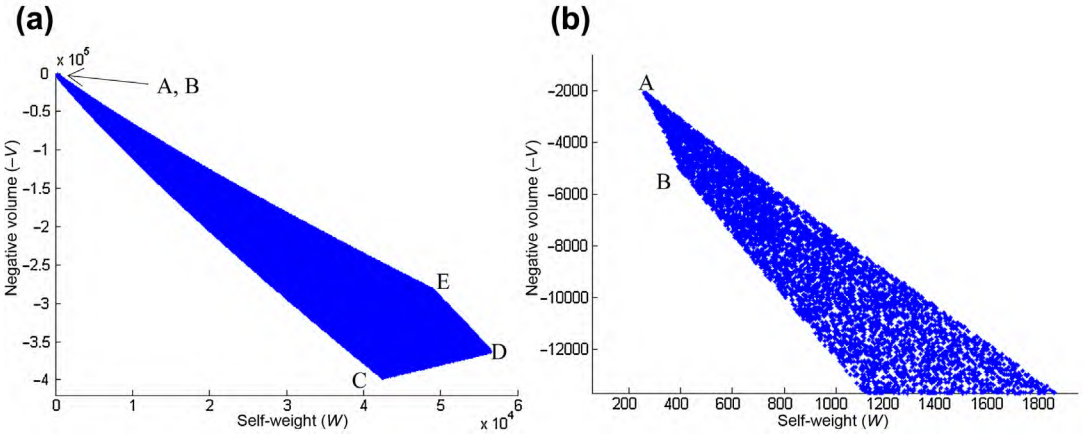


FIGURE 16.27

Pareto optimal of the cooperative game: (a) feasible set enclosed by points A to E and (b) zoomed-in view near points A and B.

$$5t \leq R \leq 8t \tag{16.95}$$

$$2.5 \leq R \leq 35.55 \text{ in.} \tag{16.96}$$

The solution graphs are shown in Figure 16.27, and the MATLAB script is given in Appendix A (at the book's companion site). Note that only solutions corresponding to feasible set are plotted. Curve ABC represents the Pareto solution, while the Nash solution in the objective space is curve BC, which is a subset of the Pareto front.

Finally, point B is the solution to the sequential game with player W as the leader; point C is the solution to the sequential game with player V as the leader.

16.6.2.5 Summary on Game Theory as Design Tool

Employing game theory to support engineering design broadens the scope of the type and scenarios of the design problems that can be formulated and solved. For a cooperative game, Pareto optimal and relevant solution techniques for solving multiobjective optimization problems have been well developed, as will be discussed in Chapter 19. However, formulating a design problem as a strategy form game or a sequential game that supports multiple designers or decentralized design teams offers a plausible approach to solving practical design scenarios that have not been fully explored. Although game theory offers a plausible approach for support of engineering design, more research work needs to be done, such as in developing a systematic approach for formulating and decomposing a design problem into a set of subproblems to be solved by a group of design teams. A practical and viable design tool based on game theory is intriguing and is highly anticipated in the design community.

16.7 SUMMARY

In this chapter, we focused on decision making in support of engineering design. We discussed two conventional methods, the decision matrix and decision tree, which are effective and powerful in

support of high-level decision making. Risk and uncertainty are the two major issues that decision makers must confront. We reviewed a few basics of decision theory that support decision making under risk and uncertainty. We also discussed two decision theories, utility theory and game theory, which were explored to support engineering design. Although not matured yet, design methods based on the theories offer significant advantages over conventional decision making and optimization techniques.

One of our goals in this chapter was to introduce the design theory that led to the development of decision theories and decision-based design. We touched a bit on the topic by presenting two design examples that used utility theory and game theory as design tools, respectively. We hope the discussion provided serves as a gateway to those who are interested in broadening their view through learning new ideas and methods, and for those who are interested in entering the research field of design theory and methods.

We believe this chapter serves its purpose as a prelude to the subject of design theory and methods. We hope this chapter offers adequate breadth and depth that help readers move on to the following chapters. The remaining chapters offer somewhat more mathematical and rigorous discussion on design methods that are well developed and being used in both academia and industry. We will go over some of the key methods, including optimization for single objective (Chapter 17) and multiobjective problems (Chapter 19). Solution techniques and numerical algorithms have been developed based on gradient-based and non-gradient-based approaches. Gradient-based optimization is practically the only viable approach for solving large-scale design problems that require substantial computational efforts. In support of gradient-based optimization, the methods require calculations of gradient information that characterizes the changes in performance measures with respect to design variables—the so-called sensitivity analysis. We discuss this important topic in Chapter 18, with a focus on structural problems.

QUESTIONS AND EXERCISES

16.1. If the weighting factors listed in Table 16.3 are revised as Strength: 2, Weight: 4, Machinability: 3, Corrosion resistance: 5, and Material cost: 3, which material among the five is desirable to be used?

16.2. A manufacturing firm is moving to absorb some short-term excess production capacity at one of its plants. The firm is considering a short manufacturing run for either of two new products, a coffee maker or a blender. The market for each product is known if the products can be successfully developed. However, there is some chance that it will not be possible to successfully develop them. Revenue of $1,000,000 would be realized from selling the coffee maker and revenue of $400,000 would be realized from selling the blender. Both of these amounts are net of production cost but do not include development cost. If development is unsuccessful for a product, then there will be no sales, and the development cost will be totally lost. Development cost would be $100,000 for the coffee maker and $10,000 for the blender. Suppose that the probability of development success is 0.5 for the coffee maker and 0.8 for the blender. Sketch a decision tree diagram to describe the problem and calculate the payoffs to help the firm make an adequate decision.

16.3. The following payoff table was developed. Let $P(S1) = 0.25$, $P(S2) = 0.55$, and $P(S3) = 0.20$. Compute the expected monetary value for each of the alternatives. What decision would you recommend?

State of Nature			
Alternative	***S*1**	***S*2**	***S*3**
A1	\$50	\$65	\$120
A2	\$85	\$45	\$80
A3	\$70	\$90	\$105

16.4. An agriculture manufacturing company has seen its business expand to the point where it needs to increase production beyond its existing capacity. It has narrowed the alternatives to two approaches to increase the maximum production capacity:

a. Expansion, at a cost of \$8 million, or

b. Modernization at a cost of \$5 million

Both approaches would require the same amount of time for implementation. Management believes that over the required payback period, demand will either be high or moderate. Because high demand is considered to be somewhat less likely than moderate demand, the probability of high demand has been estimated at 0.35. If the demand is high, expansion would gross an estimated additional \$12 million but modernization only an additional \$6 million, due to lower maximum production capability. On the other hand, if the demand is moderate, the comparable figures would be \$7 million for expansion and \$5 million for modernization. Calculate the EMV for each course of action, (a) and (b).

16.5. A steakhouse is contemplating opening a new restaurant on Main Street. It has three different models, each with a different seating capacity. They estimate that the average number of customers per hour will be 80, 120, or 160. The payoff table (profits) for the three models is developed as shown in the table below,

Average Number of Customers Per Hour			
	***S*1 = 80**	***S*2 = 120**	***S*3 = 160**
Model A	\$10,000	\$18,000	\$24,000
Model B	\$6000	\$16,000	\$12,000
Model C	\$3000	\$19,000	\$28,000

The probabilities for the states of nature are $P(S1) = 0.2$, $P(S2) = 0.35$, and $P(S3) = 0.45$. Calculate the expected value for each decision.

16.6. On a busy highway in an urban area, speeding is common. A speeding ticket costs an amount of $t > 0$. Some people drive aggressively and drive over the speed limit. There is a probability $p > 0$ of being caught, which leads to a fine $f > t$. In order to slow down the drivers on the busy highway, there are two possible concepts: doubling the fine f or doubling the patrols (i.e., the

probability p). Assuming that the drivers are risk-averse, risk-neutral, and risk-prone, which is the better concept for these three kinds of drivers respectively—double the fine or double the patrols?

16.7. Jeff is an e-trader. Suppose that Jeff's utility as a function of the money he has in his trading account, x, is given by $U(x) = \ln x$ (the natural logarithm of x).

a. Is Jeff risk averse? Explain why or why not.

b. Jeff now has \$10,000 and two possible decisions. For decision 1, he loses \$500 for certain (by paying the required fees to keep trading). For decision 2, he loses \$0 with probability 0.9 and loses \$5000 with probability 0.1. Which decision maximizes the expected utility of his money in the account?

16.8. Formulate the unconstrained beam design problem, as discussed in Section 16.6.1.1, as a strategy form game. Find the Nash equilibrium of this problem.

16.9. Solve the beam design problem shown in Section 16.6.1.1. In this case, instead of a cantilever beam, assume that the beam is clamped at both ends. Solve this unconstrained problem for the following cases:

Case 1: The designer is risk neutral with preferences $k_w = 0.5$, $k_z = 0.5$.

Case 2: The designer is risk averse for the weight attribute and is risk prone for the displacement attribute with preferences $k_w = 0.5$, $k_z = 0.5$.

Case 3: The designer is risk neutral with preferences $k_w = 0.8$, $k_z = 0.2$.

Compare the results of the three cases and make observations in terms of the influences of attitude toward risk and preferences to the solutions.

REFERENCES

Akao, Y., 2004. QFD: Quality Function Deployment – Integrating Customer Requirements into Product Design. Productivity Press, New York.

Aliprantis, C.D., Chakrabarti, S.K., 2006. Game theory in decision-making. In: Lewis, K.E., Chen, W., Schmidt, L.C. (Eds.), Decision Making in Engineering Design. ASME Press, New York, pp. 245–264.

Bernoulli, D., January 1954. Exposition of a new theory on the measurement of risk. Econometrica (The Econometric Society) (Dr. Sommer L, Trans.) 22 (1), 22–36 (originally published in 1738).

Eatas, A., Jones, J.C., 1996. The Engineering Design Process, second ed. John Wiley and Sons, Inc, New York.

Hazelrigg, G.A., 1996. Systems Engineering: An Approach to Information-Based Design. Prentice Hall, Upper Saddle River, NJ.

Huang, G.Q., 1996. In: Huang, G.Q. (Ed.), Design for X Concurrent Engineering Imperatives. Springer, Netherlands.

Hyman, B., 2003. Fundamentals of Engineering Design, second ed. Pearson Education, Inc, Upper Saddle River, NJ.

Iyer H.V. Tang X. Krishnamurty S. (1999) Constraint Handling and Iterative Attribute Method Building in Decision-Based Engineering Design., ASME Design Engineering Technical Conference DETC99/DAC-8582: Las Vegas, NV.

Lewis, K.E., Chen, W., Schmidt, L.C., 2006. Decision Making in Engineering Design. ASME Press, New York.

Lochner, R.H., Matar, J.E., 1990. Designing for Quality: An Introduction to the Best of Taguchi and Western Methods of Statistical Experimental Design. Springer, the Netherlands.

Martin, R., 2004. The st. Petersburg paradox. In: Zalta, E.N. (Ed.), The Stanford Encyclopedia of Philosophy. Fall 2004 ed. Stanford University, Stanford, CA.

National Science Foundation, 1996. Research Opportunities in Engineering Design. In: NSF Strategic Planning Workshop Final Report National Science Foundation: Washington, DC.

North, D.W., 1968. A tutorial introduction to decision theory. In: IEEE Transactions on Systems Science and Cybernetics SSC-4, 3, pp. 200–210.

Park, S.H., Antony, J., 2008. Robust Design for Quality Engineering and Six Sigma. World Scientific, Singapore; Hackensack, NJ.

Shtub, A., Brad, J.F., Globerson, S., 1994. Project Management: Engineering, Technology, and Implementation. Prentice-Hall, Inc, Englewood Cliffs, NJ.

Suh, N.P., 1990. The Principles of Design. Oxford University Press, Oxford.

Voland, G., 2004. Engineering by Design, second ed. Prentice Hall, Upper Saddle River, NJ.

von Neumann, J., Morgenstern, O., 1947. Theory of Games and Economic Behavior, second ed. Princeton University Press, Princeton, NJ.

CHAPTER

DESIGN OPTIMIZATION 17

CHAPTER OUTLINE

17.1 Introduction 910
17.2 Optimization Problems 913
17.2.1 Problem Formulation 913
17.2.2 Problem Solutions 915
17.2.3 Classification of Optimization Problems 916
17.2.4 Solution Techniques 917
17.3 Optimality Conditions 918
17.3.1 Basic Concept of Optimality 919
17.3.1.1 Functions of a Single Variable 919
17.3.1.2 Functions of Multiple Variables 920
17.3.2 Basic Concept of Design Optimization 923
17.3.3 Lagrange Multipliers 924
17.3.4 Karush–Kuhn–Tucker Conditions 927
17.4 Graphical Solutions 930
17.4.1 Linear Programming Problems 931
17.4.2 Nonlinear Programming Problems 933
17.5 Gradient-Based Approach 936
17.5.1 Generative Method 936
17.5.2 Search Methods 937
17.5.3 Gradient-Based Search 939
17.5.3.1 Steepest Descent Method 940
17.5.3.2 Conjugate Gradient Method 943
17.5.3.3 Quasi-Newton Method 944
17.5.3.4 The BFGS Method 944
17.5.4 Line Search 946
17.5.4.1 Concept of Line Search 946
17.5.4.2 Secant Method 948
17.6 Constrained Problems* 949
17.6.1 Basic Concept 950
17.6.2 ε-Active Strategy 952
17.6.3 The Sequential Linear Programming Algorithm 952
17.6.4 The Sequential Quadratic Programming Algorithm 956

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00017-X
Copyright © 2015 Elsevier Inc. All rights reserved.

17.6.5 Feasible Direction Method 957
17.6.6 Penalty Method 962
17.7 Non-Gradient Approach* 964
17.7.1 Genetic Algorithms 965
17.7.1.1 Basic Concepts 965
17.7.1.2 Design Representation 965
17.7.1.3 Selection 966
17.7.1.4 Reproduction Process and Genetic Operations 966
17.7.1.5 Solution Process 968
17.7.2 Simulated Annealing 968
17.7.2.1 Basic Concept 969
17.7.2.2 Solution Process 970
17.8 Practical Engineering Problems 970
17.8.1 Tool Integration for Design Optimization 971
17.8.2 Interactive Design Process 975
17.8.2.1 Sensitivity Display 975
17.8.2.2 What-if Study 976
17.8.2.3 Trade-off Determination 977
17.9 Optimization Software 978
17.9.1 Optimization in CAD 978
17.9.2 Optimization in FEA 979
17.9.3 Special-purpose Codes 980
17.10 Case Studies 980
17.10.1 Sizing Optimization of Roadwheel 981
17.10.1.1 Geometric Modeling and Design Parameterization 981
17.10.1.2 Analysis Model 981
17.10.1.3 Performance Measures 983
17.10.1.4 Design Sensitivity Results and Display 983
17.10.1.5 What-if Study 984
17.10.1.6 Trade-off Determination 985
17.10.1.7 Design Optimization 986
17.10.1.8 Postoptimum Study 986
17.10.2 Shape Optimization of the Engine Connecting Rod 986
17.10.2.1 Geometric and Finite Element Models 988
17.10.2.2 Design Parameterization and Problem Definition 989
17.10.2.3 Design Optimization 990
17.11 Tutorial Example: Simple Cantilever Beam 991
17.11.1 Using SolidWorks Simulation 993
17.11.2 Using Pro/MECHANICA Structure 995
17.12 Summary 996
Questions and Exercises 996
References 999

In this chapter, we discuss design optimization—one of the mainstream methods in support of engineering design. In design optimization, we minimize (or maximize) an objective function subject to performance constraints by varying a set of design variables, such as part dimensions, material properties, and so on. Usually we deal with one objective function in carrying out the so-called single-objective optimization problem, which is the subject of this chapter. Often, there are more than one objective functions to be minimized simultaneously. Such problems are called multi-objective (or multi-criteria) optimization (MOO) problems, which will be discussed in Chapter 19. The objective function and performance constraints are usually extracted from or defined for a physical problem of a single engineering discipline. For example, if we are designing a load-bearing component, in which we minimize the structural weight and subject to constraints characterizing structural strength performance, we are solving a structural optimization problem that is single disciplinary. In many situations, especially in the context of e-Design, we are dealing with designing a product or a system that involves multiple engineering disciplines, including structural, kinematic and dynamic, manufacturing, product cost, and so forth. Such design problems are called multi-disciplinary optimization (MDO), which will be briefly discussed in Chapter 19, and Projects P5 and S5 through a single-piston engine example.

The existence of optimization methods can be traced to the days of Newton, Lagrange, and Cauchy. The development of differential calculus methods for optimization was possible because of the contributions of Newton and Leibnitz to calculus. The foundations of the calculus of variations, which deals with the minimization of functions, were laid by Bernoulli, Euler, Lagrange, and Weistrass. The method of optimization for constrained problems, which involve the addition of unknown multipliers, became known by the name of its inventor, Lagrange. Cauchy made the first application of the steepest descent method to solve unconstrained optimization problems. By the middle of the twentieth century, high-speed digital computers made implementation of the numerical optimization techniques possible and stimulated further research on newer methods. Some major developments include the work of Kuhn and Tucker in the 1950s on the necessary and sufficient conditions for the optimal solution of programming problems, which laid the foundation for later research in nonlinear programming. Early development focused on gradient-based algorithms that employ gradient information of the objective and constraint functions in searching for optimal solutions. In the late twentieth century, non-gradient approaches, such as simulated annealing, genetic algorithms, and neural network methods, representing a new class of mathematical programming techniques, came into prominence. Today, optimization techniques are widely employed to support engineering design. The applications for mechanical system or component designs include car suspensions for durability, car bodies for noise/vibration/harshness, pumps and turbines for maximum efficiency, and so forth. There are plenty of applications beyond mechanical designs, such as optimal production planning, controlling, and scheduling; optimum pipeline networks for process industry; controlling the waiting and idle times in production lines to reduce the cost of production; optimum design of control systems; among many others.

This chapter offers an introductory discussion to the single-objective optimization problems. We provide a brief introduction to design optimization for those who are not familiar with the subject. Basic concepts include problem formulation, optimality conditions, and graphical solutions using simple problems for which optimal solutions can be found analytically. In addition, we discuss both linear and nonlinear programming and offer a mathematical basis for design problem formulation and solutions. We include both gradient-based and non-gradient approaches for solving optimization

problems. For those who are interested in learning more about optimization or are interested in entering this technical area for research, there are several excellent references in the literature (Vanderplaats, 2005; Rao, 2009; Arora, 2012).

Most of the solution techniques require a large amount of evaluation for the objective and constraint functions. The disciplinary or physical models that are employed for function evaluations are often very complex and must be solved numerically (e.g., using finite element methods). It can take a significant amount of computation time for a single function evaluation. As a result, the solution process for an optimization problem can be extremely time-consuming. Therefore, in this chapter, readers should see clearly the limitations of the non-gradient approaches in terms of the computational efforts for large-scale design problems. The gradient-based approaches are more suitable to the typical problems in the context of e-Design.

In addition to offering optimization concept and solution techniques, we use functions provided in MATLAB to solve example problems that are beyond hand calculations. MATLAB scripts developed for numerical examples are provided for reference on the book's companion website, http://booksite.elsevier.com/9780123820389. Moreover, we offer a discussion on the practical aspects of carrying out design optimization for practical engineering problems, followed by a brief review on commercial software tools in the hope of ensuring that readers are aware of them and their applicability to engineering applications. We include two case studies to illustrate the technical aspects of performing design optimization using commercial computer-aided design (CAD) and computer-aided engineering (CAE) software. In addition, a simple cantilever beam example, modeled in both Pro/MECHANICA Structure and SolidWorks Simulation, is offered to provide step-by-step details in using the respective software to carry out design optimization. Example files can be found on the book's companion website, http://booksite.elsevier.com/9780123820389. Detailed instructions on using these models and steps for carrying out optimization are given in Projects P5 and S5.

Overall, the objectives of this chapter are (1) to provide basic knowledge in optimization and help readers understand the concept and solution techniques, (2) to familiarize readers with practical applications of the optimization techniques and be able to apply adequate techniques for solving optimization problems using MATLAB, (3) to introduce readers to popular optimization software that is commercially available and typical practical engineering problems in case studies, and (4) to help readers become familiar with the optimization capabilities provided in Pro/MECHANICA Structure and SolidWorks Simulation for basic applications.

We hope this chapter helps readers to learn the basics of design optimization, become acquainted with the optimization subject in general, and be able to move on to the follow-up chapters to gain an in-depth knowledge of the broad topics of design methods.

17.1 INTRODUCTION

Many engineering design problems can be formulated mathematically as single-objective optimization problems, in which one single objective function is to be minimized (or maximized) subject to a set of constraints derived from requirements in, for example, product performance or physical sizes. As a simple example, we design a beer can for a maximum volume with a given amount of surface area, as shown in Figure 17.1(a). The geometry of the can is simplified as the cylinder shown in Figure 17.1(b) with two geometric dimensions, radius r and height h. The volume and surface area of the can

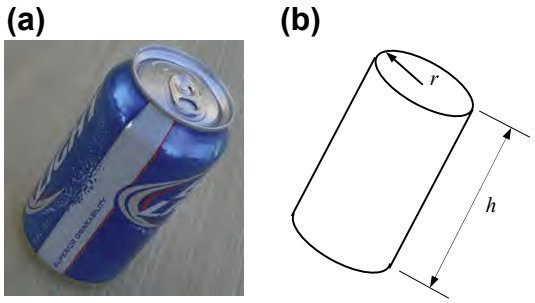


FIGURE 17.1

Beer can design: (a) beer can and (b) beer can simplified as a cylinder.

are $V = \pi r^2 h$ and $A = 2\pi r(r + h)$, respectively. The can design problem can be formulated mathematically as follows:

$$\text{Maximize:} \quad V(r, h) = \pi r^2 h \tag{17.1a}$$

$$\text{Subject to:} \quad A(r, h) = \pi r(r + 2h) = A_0 \tag{17.1b}$$

$$0 < r, \quad 0 < h \tag{17.1c}$$

where A_0 is the given amount of surface area. In this case, $V(r, h)$ is the objective function to be maximized, and $A(r, h) = 2\pi r(r + h) = A_0$ is the constraint, or more precisely, an equality constraint to be satisfied. The radius r and height h are design variables. Certainly, both r and h must be greater than 0.

Solving the beer can design problem is straightforward because both the objective and constraint functions are expressed explicitly in term of the design variables, r and h. For example, from Eq. 17.1b, we have

$$r = -h + \sqrt{h^2 + \frac{A_0}{\pi}}$$

Bringing this equation back to Eq. 17.1a, we have

$$V = \pi h \left(-h + \sqrt{h^2 + \frac{A_0}{\pi}} \right)^2 \tag{17.2}$$

We hence converted a constrained optimization problem of two design variables to an unconstrained problem of single design variable h, which is much easier to solve. How do we solve Eq. 17.2 to find the optimal solution, in this case maximum volume of the beer can? One approach is to graph the volume function in terms of design variable h. For example, if the area is given as $A_0 = \pi$, Eq. 17.2 can be graphed as shown in Figure 17.2, for example, using MATLAB. Readers are referred to the book's companion website, http://booksite.elsevier.com/9780123820389 (Script 17.1) to find the script that graphs the curve. From the graph, we can easily see that when the height h is about 1.2, volume reaches its maxima around 4.8.

The maximum of Eq. 17.2 can also be found by finding the solution of the derivative of Eq. 17.2, $dV/dh = 0$, and checking if the solution h satisfies $d^2V/dh^2 < 0$ (or if the objective function is concave),

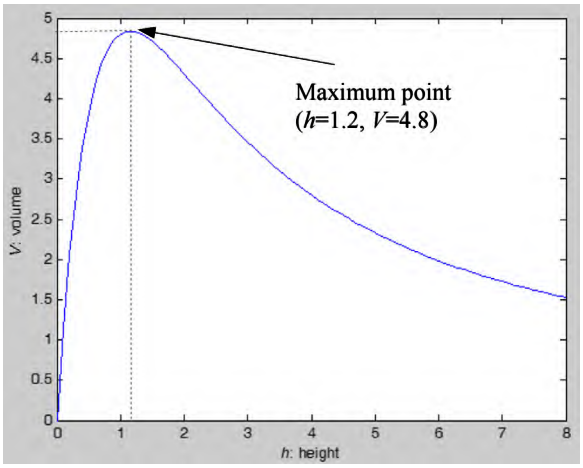


FIGURE 17.2

Graph of volume V in height h for the solution of the beer can example.

as we learned in calculus. As seen in Figure 17.2, at $h = 1.2$, we have $dV/dh = 0$. Also, the function curve is concave in the neighborhood of $h = 1.2$. Hence, the rate of slope change is negative; that is, $d^2V/dh^2 < 0$.

The beer can design problem represents the simplest kind of optimization problem, in which both the objective and constraint are explicit functions of design variables. In most engineering problems, objective and constraint functions are too complex to be expressed in terms of design variables explicitly. In many cases, they have to be evaluated using numerical methods, for example, finite element methods. When the physical model (created in computer) to be solved for the objective and constraint function evaluations is large, the solution process for an optimization problem can be extremely time-consuming. This is especially true for multidisciplinary problems, in which performance constraints are evaluated through intensive computations of multiple physical models involved in characterizing the physical behavior of the product. As a result, many methods and algorithms are developed to support design optimization by reducing computation time through minimizing the number of function evaluations.

In this chapter, we use simple and analytical examples to illustrate the optimization concept and some of the most popular solution techniques. When you review the concept and solution methods, please keep in mind that the objective and constraint functions are not necessarily expressed in terms of design variables explicitly. We discuss optimization problem formulation in Section 17.2, and then we introduce optimality conditions in Section 17.3. We include three basic solution approaches: the graphical methods in Section 17.4, and the gradient-based methods for constrained and unconstrained problems in Sections 17.5 and 17.6, respectively. Two popular solution techniques using a non-gradient approach, genetic algorithm and simulated annealing, are briefly discussed in Section 17.7. In Section 17.8, we discuss practical aspects of solving engineering optimization problems, followed by a short review on optimization software in Section 17.9. In Section 17.10, we include two case studies, followed by a tutorial example in Section 17.11.

17.2 OPTIMIZATION PROBLEMS

An optimization problem is a problem in which certain parameters (design variables) need to be determined to achieve the best measurable performance (objective function) under given constraints. In Section 17.1, we introduced the basic idea of design optimization using a very simple beer can example. The design problem is straightforward to formulate, analytical expressions are available for the objective and constraint functions, and the optimal solution is obtained graphically. Although the simple problem illustrates the basic concepts of design optimization, in reality, design problems are much more involved in many aspects, including problem formulation, solutions, and results interpretation.

17.2.1 PROBLEM FORMULATION

In general, a single-objective optimization problem can be formulated mathematically as follows:

$$\text{Minimize: } f(\mathbf{x}) \tag{17.3a}$$

$$\text{Subject to: } g_i(\mathbf{x}) \leq 0, \quad i = 1, m \tag{17.3b}$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, p \tag{17.3c}$$

$$x_k^{\ell} \leq x_k \leq x_k^{u}, \quad k = 1, n \tag{17.3d}$$

where $f(\mathbf{x})$ is the objective function or goal to be minimized (or maximized); $g_i(\mathbf{x})$ is the ith inequality constraint; m is the total number of inequality constraint functions; $h_j(\mathbf{x})$ is the jth equality constraint; p is the total number of equality constraints; $\mathbf{x}$ is the vector of design variables, $\mathbf{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$; n is the total number of design variables; and x_k^{ℓ} and x_k^{u} are the lower and upper bounds of the kth design variable x_k, respectively. Note that Eq. 17.3d is called side constraints. Equation (17.3) can also be written in a shorthand version as

$$\underset{\mathbf{x} \in S}{\text{Minimize}} \quad \mathrm{f}(\mathbf{x}) \tag{17.4}$$

in which S is called a feasible set or feasible region, defined as

$$S = \left\{ \mathbf{x} \in R^n | g_i(\mathbf{x}) \leq 0, \quad i = 1, m; \;\; h_j(\mathbf{x}) = 0, \quad j = 1, p; \;\; \text{and } x_k^{\ell} \leq x_k \leq x_k^{u}, \quad k = 1, n \right\} \tag{17.5}$$

Note that not all design problems can be formulated mathematically like those in Eq. 17.3 (or Eqs 17.4 and 17.5). However, whenever possible, readers are encouraged to formulate a design problem like the above in order to proceed with solution techniques for solving the design problem.

There are several steps for formulating a design optimization problem. First, the designer or the design team must develop a problem statement. What is the designer trying to accomplish? Is there a clear set of criteria or metrics to determine if the design of the product is successful at the end? Next, the designer or team must collect data and information relevant to the design problem. Is all the information needed to construct and solve the physical models of the design problem available? For example, if the design problem involves structural analysis, is the external load determined accurately?

After the above steps are completed, the designer or team faces two important tasks: design problem formulation and physical modeling.

Design problem formulation transcribes a verbal description (usually qualitative) into a quantitative statement in a mathematical form that defines the optimization problem, like that of Eq. 17.3. This task involves converting qualitative design requirements into quantitative performance measures and identifying objective function (or functions) that determine the performance or outcome of the physical system, such as costs, weight, power output, and so forth. In the meantime, the team identifies the performance constraints that the design must satisfy in accordance with the problem statement or functional and physical requirements identified at the beginning. With the objective and constraint functions identified, the team finds dimensions among other parameters (such as material properties) that largely influence the performance measures and chooses them as design variables with adequate upper or lower bounds. Performance measures are those from which both objective and performance constraints are chosen. Essentially, the goal is to formulate a design problem mathematically, as shown in Eq. 17.3.

The physical modeling involves the construction of mathematical models or equations that describe the physical behavior of the system being designed. Note that, in general, a physical problem is too complex to analyze and must be simplified so that it can be solved either analytically or numerically.

Next, we use the cross bar of the traffic light shown in Figure 17.3 to further illustrate the steps in formulating a design optimization problem. In this project, the goal is to design a structurally strong cross bar for the traffic light with a minimum cost. We have collected information needed for the design of the cross bar, including material to be used and its mechanical properties. Additionally, the length of the cross bar has to be 30 ft. to meet a design requirement. The cross bar can be modeled as a cantilever beam with self-weight and a point load due to the light box at the tip of the beam. Without involving detailed geometric modeling and finite element analysis (FEA), we first simplify the problem by assuming the cross bar as a straight beam with constant cross-section. We further assume the weight of the light box is significantly larger than the weight of the cross bar; therefore, the self-weight of the beam can be neglected. We also assume a solid cross-section of rectangular shape with width w and height h. The simplified cross bar is shown schematically in Figure 17.4.

Because we want the beam to be strong and yet as inexpensive as possible, we define the volume of the beam as the objective function to be minimized and we constrain the stress of the beam to be less than its yield strength. The rationale is that a beam of lesser volume consumes less material; therefore,

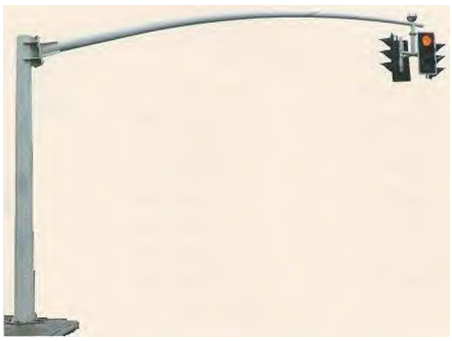


FIGURE 17.3

A traffic light employed for illustration steps of problem formulation.

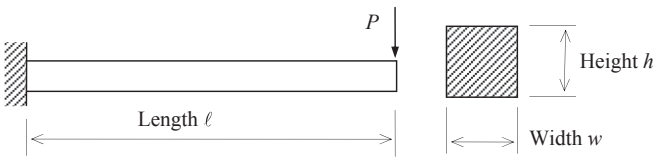


FIGURE 17.4

Cantilever beam example.

it is less expensive. Next, we pick the width and height as design variables. At this point, we need to construct a physical model with equations that govern the behavior of the beam and relate the design variables to the objective and constraint functions. For the objective function, the equation is straightforward; that is, $V(w, h) = wh\ell$. For the stress measure, we use the bending stress equation of the cantilever beam; that is, $\sigma(w, h) = \frac{6P\ell}{wh^2}$. Thereafter, the design problem of the cantilever beam example can be formulated mathematically as

$$\text{Minimize: } V(w, h) = wh\ell \tag{17.6a}$$

$$\text{Subject to: } \sigma(w, h) = \frac{6P\ell}{wh^2} - S_y \leq 0 \tag{17.6b}$$

$$w > 0, \quad h > 0 \tag{17.6c}$$

Note that in Eq. 17.6b, S_y is the material yield strength.

In general, physical modeling is not as straightforward as that shown in the cantilever beam example. Numerical simulations, such as finite element methods, are employed for the evaluation of product performance. For topics in physical modeling using numerical simulations, readers are encouraged to refer to Chapters 7 to 9. Function evaluations that support design optimization are carried out by the analysis of the physical models. If cost is involved in the design problem formulation, the readers are referred to Chapter 15 or references, such as Ostwald and McLaren (2004), Ostwald (1991), and Clark and Lorenzoni (1996), for more information.

17.2.2 PROBLEM SOLUTIONS

Once the problem is formulated, we need to solve it for an optimal solution. The solution process involves selecting a most suitable optimization technique or algorithm to find an optimal solution. In general, an optimization problem is solved numerically, in which it is required that designers understand the basic concept and the pros and cons of various optimization techniques. For problems with two or less design variables, the graphical method is an excellent choice. Some simple problems, where objective and constraint functions are written explicitly in terms of design variables, can be solved by using the necessary and sufficient conditions of the optimality. All of these solution techniques will be discussed in this chapter. Most importantly, once an optimal solution is obtained, designers must analyze, interpret, and validate the solutions before presenting the results to others.

For the cantilever beam example, we use the graphical solution technique because there are only two design variables, width w and height h of the beam cross-section. We first graph schematically the stress constraint function σ and side constraints on a plane of two axes w and h, as shown in

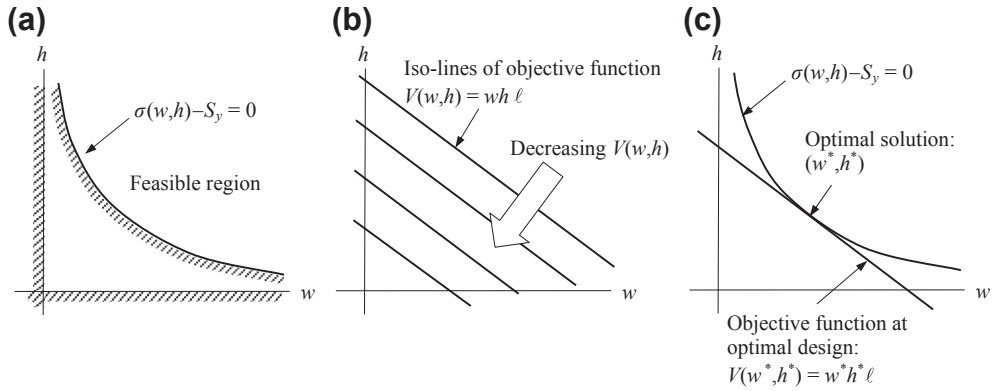


FIGURE 17.5

Cantilever beam design problem solved by using the graphical technique: (a) feasible region, (b) iso-lines of the objective function, and (c) optimal solution identified at (w^*, h^*).

Figure 17.5(a). All designs (w, h) that satisfy the constraints are called feasible designs. The set that collects all feasible designs is called a feasible set or feasible region. For this example, the feasible region S can be written as:

$$S = \{(w,h) \in R^2 \,|\, \sigma(w,h) - S_y \leq 0, w > 0, \text{ and } h > 0\} \tag{17.7}$$

Next, we plot the objective function $V(w, h)$ in Figure 17.5(b) with its iso-lines—in this case, straight lines. The iso-lines of the objective function are decreasing toward the origin of the w–h plane. It is clear that the minimum of the objective function is when the iso-line reaches the origin, in which the objective function V is zero. Certainly, this result is impossible physically. Mathematically, such a solution is called infeasible because the design is not in the feasible region. An optimal solution must be sought in the feasible region. Therefore, we graph the objective function in the feasible region as in Figure 17.5(c), in which the objective function intersects the boundary of the feasible region at (w^*, h^*) to reach its minimum $V(w^*, h^*) = w^*h^*\ell$.

17.2.3 CLASSIFICATION OF OPTIMIZATION PROBLEMS

Both the beer can and beam examples involve constraints, either equality (beer can example) or inequality (beam example). Both are called constrained problems. Occasionally, constrained problems can be converted into unconstrained problems; for example, the constrained problem of the beer can in Eq. 17.1 was converted into the unconstrained problem in Eq. 17.2. Solution techniques for constrained and unconstrained problems and other kinds of problems are different.

Optimization problems can be classified in numerous ways. First, a design problem defined in Eq. 17.3 (and as in the beer can and beam examples) is called a single-objective (or single-criterion) problem because there is one single objective function to be optimized. If a design problem involves multiple objective functions, it is called a multiobjective (or multicriterion) problem. In this case, the design goal is to minimize (or maximize) all objective functions simultaneously. We discuss multiobjective optimization in Chapter 19.

As for the constraint functions, as can be seen in the examples, there are equality and inequality constraints. An optimization problem may have only equality constraints or inequality constraints, or both. Such problems are called constrained optimization problems. If there are no constraints involved, these problems are called unconstrained problems. Furthermore, based on the nature of expressions for objective and constraint functions, optimization problems can be classified as linear, nonlinear, and quadratic programming problems. That is, if all functions are linear, such problems are called linear optimization problems and they are solved by using linear programming techniques. If one of these functions is nonlinear, they are nonlinear problems, and they are solved by nonlinear programming (NLP) techniques. This classification is extremely useful from a computational point of view because there are special methods or algorithms developed for the efficient solution of a particular class of problems. Thus, the first task a designer needs to investigate is the class of problem encountered or formulated. This will, in many cases, dictate the solution techniques to be adopted in solving the problem.

As to the types of disciplines of the physical models involved in the objective and constraint functions, there are single-disciplinary and MDO problems. Structural optimization, in which only structural performance measures, such as stress, displacement, buckling load, and natural frequency, are involved in the optimization problems, is in general single disciplinary. We will discuss more on the subject of structural design in Chapter 19. MDO usually involves structural, motion, thermal, fluid, manufacturing, and so on. We include a tutorial example in Projects P5 and S5 to illustrate some of the aspects of the topic, in addition to a case study in Chapter 19.

In the beer can and beam examples, all the design variables are permitted to take any real value (in these cases, positive real values), and the optimization problem is called a real-valued programming problem. In many problems, this may not be the case. If one of the design variables is discrete, the problems are called discrete optimization or integer programming problems. Solving discrete optimization problems is a whole lot different than solving problems with continuous design variables of real numbers. In this book, we assume design variables are continuous real numbers. Those who are interested in discrete optimization problems may refer to Kouvelis and Yu (1997) and Syslo et al. (1983) for more in-depth discussions.

Based on the deterministic nature of the variables involved, optimization problems can be classified as deterministic and stochastic programming problems. A stochastic programming problem is an optimization problem in which some or all of the parameters (design variables and/or preassigned parameters) are expressed probabilistically (nondeterministic or stochastic), such as estimates of the life span of structures that have probabilistic inputs of strength and load capacity. If all design variables are deterministic, we have deterministic optimization problems. In this chapter, we focus on deterministic programming problems. In Chapter 19, we briefly discuss stochastic programming problems.

So far, we have assumed that a single designer or single design team is working on the design problems. On some occasions, there are multiple designers or design groups making respective design decisions for the same product, especially for large-scale and complex systems. In these cases, design methods that employ game theory (discussed in Chapter 16) to aid design decision making (Vincent, 1983) are still an open topic, which is continuously being explored by the technical community.

17.2.4 SOLUTION TECHNIQUES

In general, solution techniques for optimization problems, constrained or unconstrained, can be categorized into three major groups: optimality criteria methods (also called classical methods), graphical methods, and search methods using numerical algorithms, as shown in Figure 17.6.

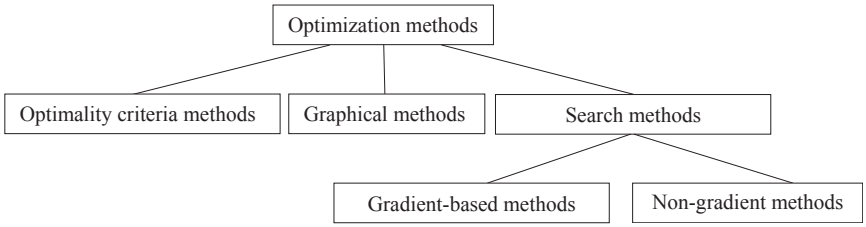


FIGURE 17.6

Classification of solution techniques for optimization problems.

The classical methods of differential calculus can be used to find the unconstrained maxima and minima of a function of several variables. These methods assume that the function is differentiable twice with respect to the design variables and the derivatives are continuous. For problems with equality constraints, the Lagrange multiplier method can be used. If the problem has inequality constraints, the Karush–Kuhn–Tucker (KKT) conditions can be used to identify the optimum point. However, these methods lead to a set of nonlinear simultaneous equations that may be difficult to solve. The classical methods of optimization are discussed in Section 17.3.

In Section 17.4, we discuss graphical solutions for solving linear and nonlinear examples. Graphical methods provide a clear picture of feasible region and iso-lines of objective functions that are straightforward in identifying optimal solutions. However, they are effective for up to two design variables, which substantially limit their applications. Note that neither classical methods nor graphical methods require numerical calculations for solutions.

The mainstream solution techniques for optimization problems are search methods involving numerical calculations that search for optimal solutions in an iterative process by starting from an initial design. Some techniques rely on gradient information (i.e., derivatives of objective and constraint functions with respect to design variables) to guide the search process. These methods are called gradient-based approaches. Other techniques follow certain rules for search optimal solutions that do not require gradient information. These are called non-gradient approaches. We provide a basic discussion on the gradient-based methods in Section 17.5 and narrow them into three major algorithms in Section 17.6, including sequential linear programming (SLP), sequential quadratic programming (SQP), and feasible direction method. We include two key algorithms of non-gradient methods in Section 17.7: genetic algorithms and simulated annealing.

17.3 OPTIMALITY CONDITIONS

A basic knowledge of optimality conditions is important for understanding the performance of the various numerical methods discussed later in the chapter. In this section, we introduce the basic concept of optimality, the necessary and sufficient conditions for the relative maxima and minima of a function, as well as the solution methods based on the optimality conditions. Simple examples are used to explain the underlying concepts. The examples will also show the practical limitations of the methods.

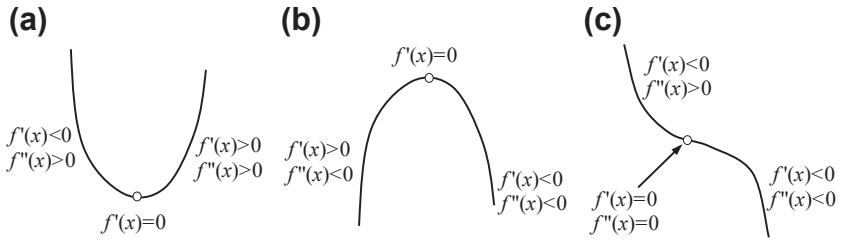


FIGURE 17.7

A stationary point may be (a) a minimum, (b) a maximum, or (c) an inflection point.

17.3.1 BASIC CONCEPT OF OPTIMALITY

We start by recalling a few basic concepts we learned in calculus regarding maxima and minima, followed by defining local and global optima; thereafter, we illustrate the concepts using functions of one and multiple variables.

17.3.1.1 Functions of a Single Variable

This section presents a few definitions for basic terms.

Stationary point: For a continuous and differentiable function $f(x)$, a stationary point x^* is a point at which the slope of the function vanishes—that is, $f'(x) = df/dx = 0$ at $x = x^*$, where x^* belongs to its domain of definition. As illustrated in Figure 17.7, a stationary point can be a minimum if $f''(x) > 0$, a maximum if $f''(x) < 0$, or an inflection point if $f''(x) = 0$ in the neighborhood of x^*.

Global and local minimum: A function $f(x)$ is said to have a local (or relative) minimum at $x = x^*$ if $f(x^*) \leq f(x^* + \delta)$ for all sufficiently small positive and negative values of δ, that is, in the neighborhood of the point x^*. A function $f(x)$ is said to have a global (or absolute) minimum at $x = x^*$ if $f(x^*) \leq f(x)$ for all x in the domain over which $f(x)$ is defined. Figure 17.8 shows the global and local optimum points of a function $f(x)$ with a single variable x.

Necessary condition: Consider a function $f(x)$ of single variable defined for $a < x < b$. To find a point of $x^* \in (a, b)$ that minimizes $f(x)$, the first derivative of function $f(x)$ with respect to x at $x = x^*$ must be a stationary point; that is, $f'(x^*) = 0$.

Sufficient condition: For the same function $f(x)$ stated above and $f'(x^*) = 0$, then it can be said that $f(x^*)$ is a minimum value of $f(x)$ if $f''(x^*) > 0$, or a maximum value if $f''(x^*) < 0$.

EXAMPLE 17.1

Find a minimum of the function $f(x) = x^2 - 2x$, for $x \in (0, 2)$.

Solution

The first derivative of $f(x)$ with respect to x is $f'(x) = 2x - 2$. We set $f'(x) = 0$, and solve for $x = 1$, which is a stationary point. This is the necessary condition for $x = 1$ to a minimum of the function $f(x)$.

We take the second derivative of $f(x)$ with respect to x, $f''(x) = 2 > 0$, which satisfies the sufficient condition of the function $f(x)$ that has a minimum at $x = 1$, and the minimum value of the function at $x = 1$ is $f(1) = -1$.

The concept illustrated above can be easily extended to functions of multiple variables. We use functions of two variables to provide a graphical illustration on the concepts.

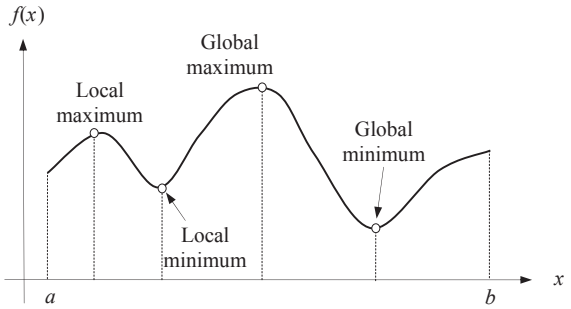


FIGURE 17.8

Global and local minimum of a function $f(x)$.

17.3.1.2 Functions of Multiple Variables

A function of two variables $f(x_1, x_2) = -(\cos^2 x_1 + \cos^2 x_2)^2$ is graphed in Figure 17.9(a). Perturbations from point $(x_1, x_2) = (0, 0)$, which is a local minimum, in any direction result in an increase in the function value of $f(\mathbf{x})$; that is, the slopes of the function with respect to x_1 and x_2 are zero at this point of local minimum. Similarly, a function $f(x_1, x_2) = (\cos^2 x_1 + \cos^2 x_2)^2$ graphed in Figure 17.9(b) has a local maximum at $(x_1, x_2) = (0, 0)$. Perturbations from this point in any direction result in a decrease in the function value of $f(\mathbf{x})$; that is, the slopes of the function with respect to x_1 and x_2 are zero at this point of local maximum. The first derivatives of the function with respect to the variables are zero at the minimum or maximum, which again is called a stationary point.

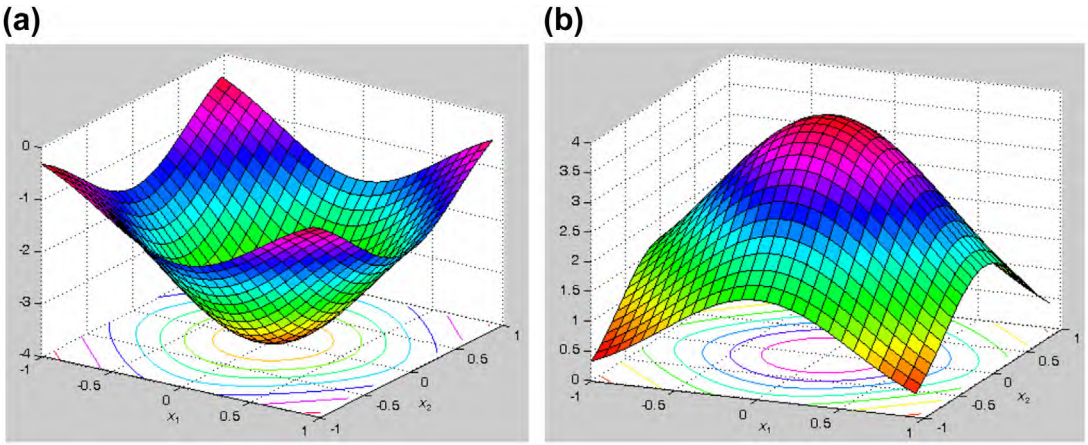


FIGURE 17.9

Functions of two variables (MATLAB Script 2 can be found in Appendix A on the book's companion website, http://booksite.elsevier.com/9780123820389): (a) $f(x_1, x_2) = -(\cos^2 x_1 + \cos^2 x_2)^2$ with a local minimum at (0, 0) and (b) $f(x_1, x_2) = (\cos^2 x_1 + \cos^2 x_2)^2$ with a local maximum at (0, 0).

Necessary condition: Consider a function $f(\mathbf{x})$ of multivariables defined for $\mathbf{x} \in R^n$, where n is the number of variables. To find a point of $\mathbf{x}^* \in R^n$ that minimizes $f(\mathbf{x})$, the gradient of the function $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}^*$ must be a stationary point; that is, $\nabla f(\mathbf{x}^*) = 0$.

The gradient of a function of multivariables is defined as

$$\nabla f(\mathbf{x}) \equiv \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]^{\mathrm{T}} \tag{17.8}$$

Geometrically, the gradient vector is normal to the tangent plane at a given point $\mathbf{x}$, and it points in the direction of maximum increase in the function. These properties are quite important; they will be used in developing optimality conditions and numerical methods for optimum design. In Example 17.2, the gradient vector for a function of two variables is calculated for illustration purposes.

EXAMPLE 17.2

A function of two variables is defined as

$$f(x_1, x_2) = x_2 e^{-x_1^2 - x_2^2} \tag{17.9a}$$

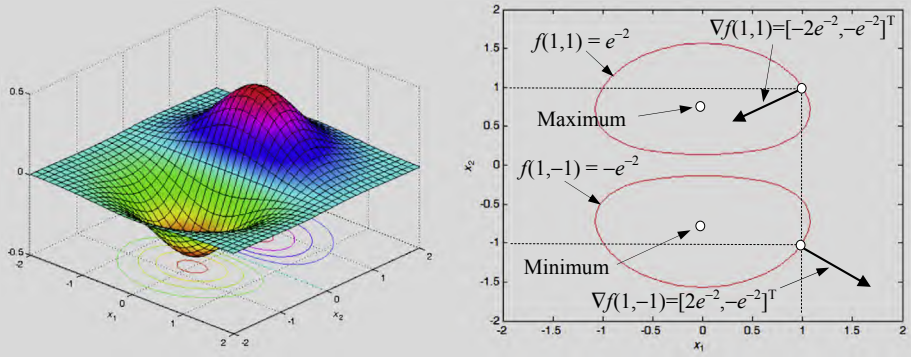
which is graphed in MATLAB shown below (left). The MATLAB script for the graph can be found on the book's companion website, http://booksite.elsevier.com/9780123820389 (Script 11.3). Calculate the gradient vectors of the function at $(x_1, x_2) = (1, 1)$ and $(x_1, x_2) = (1, -1)$.

Solution

From Eq. 17.8, the gradient vector of the function $f(x_1, x_2)$ is

$$\nabla f(x_1, x_2) = \left[-2x_1 x_2 e^{-x_1^2 - x_2^2}, e^{-x_1^2 - x_2^2} - 2x_2^2 e^{-x_1^2 - x_2^2}\right]^{\mathrm{T}} \tag{17.9b}$$

At $(x_1, x_2) = (1, 1)$, $f(1, 1) = e^{-2} = 0.1353$, and $\nabla f(1, 1) = [-2e^{-2}, -e^{-2}]^{\mathrm{T}}$; and at $(x_1, x_2) = (1, -1)$, $f(1, -1) = -e^{-2} = -0.1353$, and $\nabla f(1, -1) = [2e^{-2}, -e^{-2}]^{\mathrm{T}}$. The iso-lines of $f(1, 1)$ and $f(1, -1)$ as well as the gradient vectors at (1, 1) and (1, −1) are shown in the figure below (right). In this example, gradient vector at a point $\mathbf{x}$ is perpendicular to the tangent line at $\mathbf{x}$, and the vector points in the direction of maximum increment in the function value. The maximum and minimum of the function are shown for clarity.



Sufficient condition: For the same function $f(\mathbf{x})$ stated above, let $\nabla f(\mathbf{x}^*) = 0$, then $f(\mathbf{x}^*)$ has a minimum value of $f(\mathbf{x})$ if its Hessian matrix defined in Eq. 17.10 is positive-definite.

$$\mathbf{H} = \nabla^2 f = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right] = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdots & \cdots & & \cdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}_{n \times n} \tag{17.10}$$

where all derivatives are calculated at the given point $\mathbf{x}^*$. The Hessian matrix is an $n \times n$ matrix, where n is the number of variables. It is important to note that each element of the Hessian is a function in itself that is evaluated at the given point $\mathbf{x}^*$. Also, because $f(\mathbf{x})$ is assumed to be twice continuously differentiable, the cross partial derivatives are equal; that is,

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}; \quad i, j = 1, n \tag{17.11}$$

Therefore, the Hessian is always a symmetric matrix. The Hessian matrix plays a prominent role in exploring the sufficiency conditions for optimality.

Note that a square matrix is positive-definite if (a) the determinant of the Hessian matrix is positive (i.e., $|\mathbf{H}| > 0$) or (b) all its eigenvalues are positive. To calculate the eigenvalues λ of a square matrix, the following equation is solved:

$$|\mathbf{H} - \lambda \mathbf{I}| = 0 \tag{17.12}$$

where $\mathbf{I}$ is an identity matrix of $n \times n$.

EXAMPLE 17.3

A function of three variables is defined as

$$f(x_1, x_2, x_3) = x_1^2 + 2x_1x_2 + 2x_2^2 + x_3^2 - 2x_1 + x_2 + 8 \tag{17.13a}$$

Calculate the gradient vector of the function and determine a stationary point, if it exists. Calculate a Hessian matrix of the function f, and determine if the stationary point found gives a minimum value of the function f.

Solution

We first calculate the gradient of the function and set it to zero to find the stationary point(s), if any:

$$\nabla f(x_1, x_2, x_3) = [2x_1 + 2x_2 - 2, 2x_1 + 4x_2 + 1, 2x_3]^{\mathrm{T}} \tag{17.13b}$$

Setting Eq. 17.13b to zero, we have $\mathbf{x} = [2.5, -1.5, 0]^{\mathrm{T}}$, which is the only stationary point. Now, we calculate the Hessian matrix:

$$\mathbf{H} = \nabla^2 f = \begin{bmatrix} 2 & 2 & 0 \\ 2 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{17.13c}$$

EXAMPLE 17.3—cont'd

which is positive-definite because

$$|\mathbf{H}| = \begin{vmatrix} 2 & 2 & 0 \\ 2 & 4 & 0 \\ 0 & 0 & 1 \end{vmatrix} = 8 - 4 = 4 > 0 \tag{17.13d}$$

or

$$|\mathbf{H} - \lambda\mathbf{I}| = \begin{vmatrix} 2-\lambda & 2 & 0 \\ 2 & 4-\lambda & 0 \\ 0 & 0 & 1-\lambda \end{vmatrix} = (2-\lambda)(4-\lambda)(1-\lambda) - 4(1-\lambda) = 0 \tag{17.13e}$$

Solving Eq. 17.13e, we have $\lambda = 1$, 0.7639 and 5.236, which are all positive. Hence, the Hessian matrix is positive-definite; therefore, the stationary point $\mathbf{x}^* = [2.5, -1.5, 0]^T$ is a minimum point, at which the function value is $f(\mathbf{x}^*) = 4.75$.

17.3.2 BASIC CONCEPT OF DESIGN OPTIMIZATION

For an optimization problem defined in Eq. 17.3, we find design variable vector $\mathbf{x}$ to minimize an objective function $f(\mathbf{x})$ subject to the inequality constraints $g_i(\mathbf{x}) \leq 0$, $i = 1$ to m, the equality constraints $h_j(\mathbf{x}) = 0, j = 1$ to p, and the side constraints $x_k^{\ell} \leq x_k \leq x_k^u$, $k = 1, n$. In Eq. 17.5, we define the feasible set S, or feasible region, for a design problem as a collection of feasible designs. For unconstrained problems, the entire design space is feasible because there are no constraints. In general, the optimization problem is to find a point in the feasible region that gives a minimum value to the objective function. From a design perspective, in particular solving Eq. 17.3, we state the following terms.

Global minimum: A function $f(\mathbf{x})$ of n design variables has a global minimum at $\mathbf{x}^*$ if the value of the function at $\mathbf{x}^* \in S$ is less than or equal to the value of the function at any other point $\mathbf{x}$ in the feasible set S. That is,

$$f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall (\text{for all}) \ \ \mathbf{x} \in S \tag{17.14}$$

If strict inequality holds for all $\mathbf{x}$ other than $\mathbf{x}^*$ in Eq. 17.14, then $\mathbf{x}^*$ is called a strong (strict) global minimum; otherwise, it is called a weak global minimum.

Local minimum: A function $f(\mathbf{x})$ of n design variables has a local (or relative) minimum at $\mathbf{x}^* \in S$ if inequality of Eq. 17.14 holds for all $\mathbf{x}$ in a small neighborhood N (vicinity) of $\mathbf{x}^*$. If strict inequality holds, then $\mathbf{x}^*$ is called a strong (strict) local minimum; otherwise, it is called a weak local minimum.

Neighborhood N of point $\mathbf{x}^*$ is defined as the set of points:

$$N = \{\mathbf{x} | \mathbf{x} \in S \text{ with } ||\mathbf{x} - \mathbf{x}^*|| < \delta\} \tag{17.15}$$

for some small $\delta > 0$. Geometrically, it is a small feasible region around point $\mathbf{x}^*$, such as a sphere of radius δ for $n = 3$ (number of design variables $n = 3$).

Next, we illustrate the derivation of the necessary and sufficient conditions using Taylor's series expansion. For the time being, we assume unconstrained problems. In the next subsection, we extend the discussion to constrained problems.

Expanding the objective function $f(\mathbf{x})$ at the inflection point $\mathbf{x}^*$ using Taylor's series, we have

$$f(\mathbf{x}) = f(\mathbf{x}^*) + \nabla f(\mathbf{x}^*)^{\mathrm{T}} \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}} H(\mathbf{x}^*)\Delta\mathbf{x} + R \tag{17.16}$$

where R is the remainder containing higher-order terms in $\Delta\mathbf{x}$, and $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}^*$. We define increment $\Delta f(\mathbf{x})$ as

$$\Delta f(\mathbf{x}) = f(\mathbf{x}) - f(\mathbf{x}^*) = \nabla f(\mathbf{x}^*)^{\mathrm{T}} \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}} H(\mathbf{x}^*)\Delta\mathbf{x} + R \tag{17.17}$$

If we assume a local minimum at $\mathbf{x}^*$, then Δf must be nonnegative due to the definition of a local minimum given in Eq. 17.14; that is, $\Delta f \geq 0$.

Because $\Delta\mathbf{x}$ is small, the first-order term $\nabla f(\mathbf{x}^*)^{\mathrm{T}}\,\Delta\mathbf{x}$ dominates other terms, and therefore Δf can be approximated as $\Delta f(\mathbf{x}) = \nabla f(\mathbf{x}^*)^{\mathrm{T}}\Delta\mathbf{x}$. Note that Δf in this equation can be positive or negative depending on the sign of the term $\nabla f(\mathbf{x}^*)^{\mathrm{T}}\Delta\mathbf{x}$. Because $\Delta\mathbf{x}$ is arbitrary (a small increment in $\mathbf{x}^*$), its components may be positive or negative. Therefore, we observe that Δf can be nonnegative for all possible $\Delta\mathbf{x}$ unless

$$\nabla f(\mathbf{x}^*) = 0 \tag{17.18}$$

In other words, the gradient of the function at $\mathbf{x}^*$ must be zero. In the component form, this necessary condition becomes

$$\frac{\partial f(\mathbf{x}^*)}{\partial x_i} = 0, \quad i = 1, n \tag{17.19}$$

Again, points satisfying Eq. 17.18 or Eq. 17.19 are called stationary points.

Considering the second term on the right-hand side of Eq. 17.17 evaluated at a stationary point $\mathbf{x}^*$, the positivity of Δf is assured if

$$\Delta\mathbf{x}^{\mathrm{T}}\mathbf{H}(\mathbf{x}^*)\,\Delta\mathbf{x} > 0 \tag{17.20}$$

for all $\Delta\mathbf{x} \neq \mathbf{0}$. This is true if the Hessian $\mathbf{H}(\mathbf{x}^*)$ is a positive-definite matrix, which is then the sufficient condition for a local minimum of $f(\mathbf{x})$ at $\mathbf{x}^*$.

17.3.3 LAGRANGE MULTIPLIERS

We begin the discussion of optimality conditions for constrained problems by including only the equality constraints in the formulation in this section; that is, inequalities in Eq. 17.3b are ignored temporarily. More specifically, the optimization problem is restated as

$$\text{Minimize:} \quad f(\mathbf{x}) \tag{17.21a}$$

$$\text{Subject to:} \quad h_j(\mathbf{x}) = 0, \quad j = 1, p \tag{17.21b}$$

$$x_k^{\ell} \leq x_k \leq x_k^{u}, \quad k = 1, n \tag{17.21c}$$

The reason is that the nature of equality constraints is quite different from that of inequality constraints. Equality constraints are always active for any feasible design, whereas an inequality

constraint may not be active at a feasible point. This changes the nature of the necessary conditions for the problem when inequalities are included.

A common approach for dealing with equality constraints is to introduce scalar multipliers associated with each constraint, called Lagrange multipliers. These multipliers play a prominent role in optimization theory as well as in numerical methods, in which a constrained problem is converted into an unconstrained problem that can be solved by using optimality conditions or numerical algorithms specifically developed for them. The values of the multipliers depend on the form of the objective and constraint functions. If these functions change, the values of the Lagrange multipliers also change.

Through Lagrange multipliers, the constrained problem (with equality constraints) shown in Eq. 17.21 is converted into an unconstrained problem as

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{j=1}^{p} \lambda_j h_j(\mathbf{x}) = f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{T}} \mathbf{h}(\mathbf{x}) \tag{17.22}$$

which is called a Lagrangian function, or simply Lagrangian. If we expand the vector of design variables to include the Lagrange multipliers, then the necessary and sufficient conditions of a local minimum discussed in the previous subsection are applicable to the problem defined in Eq. 17.22.

Before discussing the optimality conditions, we defined an important term called regular point. Consider the constrained optimization problem defined in Eq. 17.21, a point $\mathbf{x}^*$ satisfying the constraint functions $\mathbf{h}(\mathbf{x}^*) = 0$ is said to be a regular point of the feasible set if the objective $f(\mathbf{x}^*)$ is differentiable and gradient vectors of all constraints at the point $\mathbf{x}^*$ are linearly independent. Linear independence means that no two gradients are parallel to each other, and no gradient can be expressed as a linear combination of the others. When inequality constraints are included in the problem definition, then for a point to be regular, gradients of all the active constraints must also be linearly independent.

The necessary condition (or Lagrange multiplier theorem) is stated next.

Consider the optimization problem defined in Eq. 17.21. Let $\mathbf{x}^*$ be a regular point that is a local minimum for the problem. Then, there exist unique Lagrange multipliers λ_j^*, $j = 1, p$ such that

$$\nabla L(\mathbf{x}^*, \boldsymbol{\lambda}^*) = \frac{\partial L(\mathbf{x}^*, \boldsymbol{\lambda}^*)}{\partial x_i} = \mathbf{0}; \quad i = 1, n \tag{17.23}$$

Differentiating the Lagrangian $L(\mathbf{x}, \boldsymbol{\lambda})$ with respect to λ_j, we recover the equality constraints as

$$\frac{\partial L(\mathbf{x}^*, \boldsymbol{\lambda}^*)}{\partial \lambda_i} = \mathbf{0}; \quad \Rightarrow h_j(\mathbf{x}^*) = 0; \quad j = 1, p \tag{17.24}$$

The gradient conditions of Eqs 17.23 and 17.24 show that the Lagrangian is stationary with respect to both $\mathbf{x}$ and $\boldsymbol{\lambda}$. Therefore, it may be treated as an unconstrained function in the variables $\mathbf{x}$ and $\boldsymbol{\lambda}$ to determine the stationary points. Note that any point that does not satisfy the conditions cannot be a local minimum point. However, a point satisfying the conditions need not be a minimum point either. It is simply a candidate minimum point, which can actually be an inflection or maximum point.

The second-order necessary and sufficient conditions, similar to that of Eq. 17.20, in which the Hessian matrix includes terms of Lagrange multipliers, can distinguish between the minimum, maximum, and inflection points. More specifically, a sufficient condition for $f(\mathbf{x})$ to have a local

minimum at $\mathbf{x}^*$ is that each root of the polynomial in ε, defined by the following determinant equation be positive:

$$\begin{vmatrix} \mathbf{L} - \mathbf{I}\varepsilon & \mathbf{G} \\ \mathbf{G} & \mathbf{0} \end{vmatrix} = 0 \tag{17.25}$$

where

$$\mathbf{L} - \mathbf{I}\varepsilon = \begin{bmatrix} L_{11} - \varepsilon & L_{12} & \ldots & L_{1n} \\ L_{21} & L_{22} - \varepsilon & \ldots & L_{2n} \\ \ldots & \ldots & & \ldots \\ L_{n1} & L_{n2} & \ldots & L_{nn} - \varepsilon \end{bmatrix}_{n \times n} \tag{17.26}$$

and

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \cdots & \cdots & & \cdots \\ g_{m1} & g_{m2} & \cdots & g_{mn} \end{bmatrix}_{m \times n} \tag{17.27}$$

Note that L_{ij} is a partial derivative of the Lagrangian L with respect to x_i and λ_j, i.e., $\boldsymbol{L}_{ij} = \frac{\partial^2 L(\mathbf{x}^*, \boldsymbol{\lambda}^*)}{\partial x_i \partial x_j}$, i, $j = 1, n$; and g_{pq} is the partial derivative of g_p with respect to x_q; i.e., $g_{pq} = \frac{\partial g_p(\mathbf{x}^*)}{\partial x_q}$, $p = 1, m$ and $q = 1, n$.

EXAMPLE 17.4

Find the optimal solution for the following problem:

$$\text{Minimize:} \quad f(\mathbf{x}) = 3x_1^2 + 6x_1x_2 + 5x_2^2 + 7x_1 + 5x_2 \tag{17.28a}$$

$$\text{Subject to:} \quad g(\mathbf{x}) = x_1 + x_2 - 5 = 0 \tag{17.28b}$$

Solution

Define the Lagrangian as

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}) = (3x_1^2 + 6x_1x_2 + 5x_2^2 + 7x_1 + 5x_2) + \lambda(x_1 + x_2 - 5)$$

Taking derivatives of $L(\mathbf{x}, \lambda)$ with respect to x_1, x_2, and λ, respectively, we have

$$\partial L/\partial x_1 = 6x_1 + 6x_2 + 7 + \lambda = 0$$

From Eq. 17.28b, $6(x_1 + x_2) = 6(5) = -7 - \lambda$. Therefore, $\lambda = -37$.

Also, $\partial L/\partial x_2 = 6x_1 + 10x_2 + 5 + \lambda = 0$. It can be also written as $6(x_1 + x_2) + 4x_2 + 5 + \lambda = 6(5) + 4x_2 + 5 - 37 = 0$. Hence $x_2 = 0.5$ and $x_1 = 4.5$.

We obtain the stationary point $\mathbf{x}^* = [4.5, 0.5]^{\mathrm{T}}$ and $\lambda^* = -37$. Next, we check the sufficient condition of Eq. 17.25; that is, for this example, we have

$$\begin{vmatrix} L_{11} - \varepsilon & L_{12} & g_{11} \\ L_{12} & L_{22} - \varepsilon & g_{21} \\ g_{11} & g_{21} & 0 \end{vmatrix} = 0$$

EXAMPLE 17.4—cont'd

in which

$$L_{11} = \left.\frac{\partial^2 L}{\partial x_1^2}\right|_{(\mathbf{x}^*,\boldsymbol{\lambda}^*)} = 6,\ L_{12} = L_{21} = \left.\frac{\partial^2 L}{\partial x_1 \partial x_2}\right|_{(\mathbf{x}^*,\boldsymbol{\lambda}^*)} = 6,\ L_{22} = \left.\frac{\partial^2 L}{\partial x_2^2}\right|_{(\mathbf{x}^*,\boldsymbol{\lambda}^*)} = 10,\ g_{11} = \left.\frac{\partial g}{\partial x_1}\right|_{(\mathbf{x}^*,\boldsymbol{\lambda}^*)} = 1,\ \text{and}$$

$g_{12} = \left.\dfrac{\partial g}{\partial x_2}\right|_{(\mathbf{x}^*,\boldsymbol{\lambda}^*)} = 1$. Hence the determinant becomes

$$\begin{vmatrix} 6-\varepsilon & 6 & 1 \\ 6 & 10-\varepsilon & 1 \\ 1 & 1 & 0 \end{vmatrix} = 6 + 6 - (10-\varepsilon) - (6-\varepsilon) = 0$$

Therefore, $\varepsilon = 2$. Because ε is positive, $\mathbf{x}^*$ and λ^* correspond to a minimum.

17.3.4 KARUSH–KUHN–TUCKER CONDITIONS

Next, we extend the Lagrange multiplier to include inequality constraints and consider the general optimization problem defined in Eq. 17.3.

We first transform an inequality constraint into an equality constraint by adding a new variable to it, called the slack variable. Because the constraint is of the form "$\leq$", its value is either negative or zero at a feasible point. Thus, the slack variable must always be nonnegative (i.e., positive or zero) to make the inequality an equality.

An inequality constraint $g_i(\mathbf{x}) \leq 0$ is equivalent to the equality constraint

$$G_i(\mathbf{x}) = g_i(\mathbf{x}) + s_i^2 = 0 \tag{17.29}$$

where s_i is a slack variable. The variables s_i are treated as unknowns of the design problem along with the design variables. Their values are determined as a part of the solution. When the variable s_i has zero value, the corresponding inequality constraint is satisfied at equality. Such an inequality is called an active (or tight) constraint; that is, there is no "slack" in the constraint. For any $s_i \neq 0$, the corresponding constraint is a strict inequality. It is called an inactive constraint, and has slack given by s_i^2.

Note that once a design point is specified, Eq. 17.29 can be used to calculate the slack variable s_i^2. If the constraint is satisfied at the point (i.e., $g_i(\mathbf{x}) \leq 0$), then $s_i^2 \geq 0$. If it is violated, then s_i^2 is negative, which is not acceptable; that is, the point is not a feasible point and hence is not a candidate minimum point.

Similar to that of Section 17.3.3, through Lagrange multipliers, the constrained problem (with equality and inequality constraints) defined in Eq. 17.3 is converted into an unconstrained problem as

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \mathbf{s}) = f(\mathbf{x}) + \sum_{i=1}^{p} \lambda_i h_i(\mathbf{x}) + \sum_{j=1}^{m} \mu_j \left(g_j(\mathbf{x}) + s_j^2\right) = f(\mathbf{x}) + \boldsymbol{\lambda}^{\mathrm{T}}\mathbf{h}(\mathbf{x}) + \boldsymbol{\mu}^{\mathrm{T}}\mathbf{G}(\mathbf{x}) \tag{17.30}$$

If we expand the vector of design variables to include the Lagrange multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, and the slack variables $\mathbf{s}$, then the necessary and sufficient conditions of a local minimum discussed in the previous subsection are applicable to the unconstrained problem defined in Eq. 17.30.

Note that derivatives of the Lagrangian L with respect to $\mathbf{x}$ and $\boldsymbol{\lambda}$ lead to Eqs 17.23 and 17.24, respectively. On the other hand, the derivatives of L with respect to $\boldsymbol{\mu}$ yield converted equality constraints of Eq. 17.29. Furthermore, the derivatives of L with respect to $\mathbf{s}$ yield

$$\mu_j s_j = 0, \quad j = 1, m \tag{17.31}$$

which is an additional necessary condition for the Lagrange multipliers of "$\leq$ type" constraints given as

$$\mu_j^* \geq 0; \quad j = 1, m \tag{17.32}$$

where μ_j^* is the Lagrange multiplier for the jth inequality constraint. Equation (17.32) is referred to as the nonnegativity of Lagrange multipliers (Arora, 2012).

The necessary conditions for the constrained problem with equality and inequality constraints defined in Eq. 17.3 can be summed up in what are commonly known as the KKT first-order necessary conditions.

Karush–Kuhn–Tucker Necessary Conditions: Let $\mathbf{x}^*$ be a regular point of the feasible set that is a local minimum for $f(\mathbf{x})$, subject to $h_i(\mathbf{x}) = 0$; $i = 1, p$; $g_j(\mathbf{x}) \leq 0$; $j = 1, m$. Then there exist Lagrange multipliers $\boldsymbol{\lambda}^*$ (a p-vector) and $\boldsymbol{\mu}^*$ (an m-vector) such that the Lagrangian L is stationary with respect to x_k, λ_i, μ_j, and s_ℓ at the point $\mathbf{x}^*$; that is:

1. Stationarity:

$$\nabla L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*, \boldsymbol{s}^*) = 0; \tag{17.33a}$$

or

$$\frac{\partial L}{\partial x_k} = \frac{\partial f}{\partial x_k} + \sum_{i=1}^{p} \lambda_i \frac{\partial h_i}{\partial x_k} + \sum_{j=1}^{m} \mu_j \frac{\partial g_j}{\partial x_k} = 0; \quad k = 1, n \tag{17.33b}$$

2. Equality constraints:

$$\frac{\partial L}{\partial \lambda_i} = 0; \quad \Rightarrow h_i(\mathbf{x}^*) = 0; \quad i = 1, p \tag{17.34}$$

3. Inequality constraints (or complementary slackness condition):

$$\frac{\partial L}{\partial \mu_j} = 0; \quad \Rightarrow g_j(\mathbf{x}^*) + s_j^2 = 0; \quad j = 1, m \tag{17.35}$$

$$\frac{\partial L}{\partial s_j} = 0; \quad \Rightarrow 2\mu_j^* s_j = 0; \quad j = 1, m \tag{17.36}$$

In addition, gradients of the active constraints must be linearly independent. In such a case, the Lagrange multipliers for the constraints are unique.

EXAMPLE 17.5

Solve the following optimization problem using the KKT conditions.

$$\text{Minimize:} \quad f(\mathbf{x}) = -2x_1 - 2x_2 + \frac{1}{2}\left(x_1^2 + x_2^2\right) \tag{17.37a}$$

$$\text{Subject to:} \quad g_1(\mathbf{x}) = 3x_1 + x_2 - 6 \leq 0 \tag{17.37b}$$

$$h_1(\mathbf{x}) = x_1 - x_2 = 0 \tag{17.37c}$$

$$0 \leq x_1, \quad 0 \leq x_2 \tag{17.37d}$$

Solution

Using Eq. 17.30, we state the Lagrangian of the problem as

$$L = -2x_1 - 2x_2 + \frac{1}{2}\left(x_1^2 + x_2^2\right) + \lambda_1(x_1 - x_2) + \mu_1\left(3x_1 + x_2 - 6 + s_1^2\right) + \mu_2\left(-x_1 + s_2^2\right) + \mu_3\left(-x_2 + s_3^2\right) \tag{17.37e}$$

Taking derivatives of the Lagrangian with respect to x_1, x_2, λ_1, μ_1, μ_2, μ_3, s_1, s_2, and s_3 and setting them to zero, we have

$$\frac{\partial L}{\partial x_1} = -2 + x_1 + \lambda_1 + 3\mu_1 - \mu_2 = 0 \tag{17.37f}$$

$$\frac{\partial L}{\partial x_2} = -2 + x_2 - \lambda_1 + \mu_1 - \mu_3 = 0 \tag{17.37g}$$

$$\frac{\partial L}{\partial \lambda_1} = x_1 - x_2 = 0 \tag{17.37h}$$

$$\frac{\partial L}{\partial \mu_1} = 3x_1 + x_2 - 6 + s_1^2 = 0 \tag{17.37i}$$

$$\frac{\partial L}{\partial \mu_2} = -x_1 + s_2^2 = 0 \tag{17.37j}$$

$$\frac{\partial L}{\partial \mu_3} = -x_2 + s_3^2 = 0 \tag{17.37k}$$

$$\frac{\partial L}{\partial s_1} = 2\mu_1 s_1 = 0 \tag{17.37l}$$

$$\frac{\partial L}{\partial s_2} = 2\mu_2 s_2 = 0 \tag{17.37m}$$

$$\frac{\partial L}{\partial s_3} = 2\mu_3 s_3 = 0 \tag{17.37n}$$

Note that, as expected, Eqs 17.37h–17.37k reduce back to the constraint equations of the original optimization problem in Eqs 17.37b–17.37d. We have a total of nine equations (Eqs 17.37f–17.37n) and nine unknowns. Not all nine equations are linear. It is not guaranteed that all nine unknowns can be solved uniquely from the nine equations. As discussed before, these equations are solved in different cases. In this example, the equality constraint must be satisfied; hence, from Eq. 17.37h, $x_1 = x_2$. Next, we make assumptions and proceed with different cases.

Continued

EXAMPLE 17.5—cont'd

Case 1: we assume that the inequality constraint, Eq. 17.37i, is active; hence, $s_1 = 0$. From the same equation, we have

$$3x_1 + x_2 - 6 = 4x_1 - 6 = 0. \quad \text{Hence } x_1 = x_2 = 1.5.$$

which implies that the side constraints are not active; hence, from Eqs 17.37j and 17.37k, $s_2 \neq 0$ and $s_3 \neq 0$, implying $\mu_2 = \mu_3 = 0$ from Eqs 17.37m and 17.37n.

Solve μ_1 and λ_1 from Eqs 17.37f and 17.37g, we have $\mu_1 = 0.25$, and $\lambda_1 = -0.25$. Then, from Eq. 17.37a, the objective function is

$$f(1.5, 1.5) = -2(1.5) - 2(1.5) + 1/2(1.5^2 + 1.5^2) = -3.75$$

Case 2: we assume that the side constraints, Eqs 17.37j and 17.37k, are active; hence, $s_2 = s_3 = 0$; and $x_1 = x_2 = 0$. From Eq. 17.37i, $s_1 \neq 0$, hence $\mu_1 = 0$ from Eq. 17.37l. There is no more assumption that can be made logically. Equations (17.37f) and (17.37g) consist of three unknowns, λ_1, μ_2, and μ_3; they cannot be solved uniquely. If we further assume $\mu_2 = 0$, then we have $\lambda_1 = 2$ and $\mu_2 = -4$. If we assume $\mu_3 = 0$, then we have $\lambda_1 = -2$ and $\mu_2 = -4$. Then, from Eq. 17.37a, the objective function is

$$f(0, 0) = -2(0) - 2(0) + 1/2(0^2 + 0^2) = 0$$

which is greater than that of Case 1.

Case 3: we assume that the side constraint, Eq. 17.37j, is active; hence, $s_2 = 0$; and $x_1 = 0$. We assume constraints (17.37i) and (17.37k) are not active; hence, $s_1 \neq 0$ and $s_3 \neq 0$; implying $\mu_1 = \mu_3 = 0$. There is no more assumption that can be made logically. Equations (17.37f) and (17.37g) consist of three unknowns, λ_1, μ_2, and x_2; they cannot be solved uniquely. If we further assume $\lambda_1 = 0$, then we have $x_2 = 2$ and $\mu_2 = -2$. Then, from Eq. 17.37a, the objective function is

$$f(0, 2) = -2(0) - 2(2) + 1/2(0^2 + 2^2) = -2$$

which is greater than that of Case 1. We may proceed with a few more cases and find possible solutions. After exhausting all cases, the solution obtained in Case 1 gives a minimum value for the objective function.

As seen in the example above, solving optimization problems using the KKT conditions is not straightforward, even for a simple problem. All possible cases must be identified and carefully examined. In addition, a sufficient condition that involves second derivatives of the Lagrangian is difficult to verify. Furthermore, for practical engineering design problems, objective and constraint functions are not expressed in terms of design variables explicitly, and taking derivatives analytically is not possible. After all, KKT conditions serve well for understanding the concept of optimality.

17.4 GRAPHICAL SOLUTIONS

Graphical methods offer means to seek optimal solutions quickly without involving numerical algorithms. Graphical visualization of the optimization problem and optimal solution in the design space enhances our understanding of the concept and numerical solution techniques to be discussed later. Because graphical methods require designers to plot the objective and constraint functions in the design space, the objective and constraint functions have to be written in terms of design variables explicitly. Moreover, graphical methods are effective only for up to two design variables. We use both linear and nonlinear programming problems to illustrate the use of the methods. In some examples, we use MATLAB plot functions to graph the objective and constraint functions for illustration purposes.

17.4.1 LINEAR PROGRAMMING PROBLEMS

Linear programming (LP) problems involve a linear objective function subject to a set of linear constraint functions. The problems are formulated as

$$\text{Minimize:} \quad f(\mathbf{x}) = a_1x_1 + a_2x_2 + \ldots + a_nx_n \tag{17.38a}$$

$$\text{Subject to:} \quad h_i(\mathbf{x}) = b_{i1}x_1 + b_{i2}x_2 + \ldots + b_{in}x_n + b_{i0} = 0, \quad i = 1, m \tag{17.38b}$$

$$g_j(\mathbf{x}) = c_{j1}x_1 + c_{j2}x_2 + \ldots + c_{jn}x_n + c_{j0} \leq 0, \quad j = 1, p \tag{17.38c}$$

$$x_k^\ell \leq x_k \leq x_k^u, \quad k = 1, n \tag{17.38d}$$

In general, solving LP problems using graphical methods involves three steps:

Step 1: Sketch constraint functions (Eqs 17.38b and 17.38c) and side constraints (Eq. 17.38d) on the design plane of two design variables, x_1 and x_2.

Step 2: Identify the feasible region, in which any point in the region satisfies the constraints.

Step 3a: Sketch iso-lines of the objective function on top of the feasible region and identify the optimal point (usually a vertex of the polygon of the feasible region on the x_1–x_2 plane), or

Step 3b: Calculate the values of the vertices of the feasible region, and plug these values into the objective function (Eq. 17.38a) to find the minimum.

Example 17.6 illustrates these steps.

EXAMPLE 17.6

Solve the following LP problem using the graphical method.

$$\text{Minimize:} \quad f(\mathbf{x}) = 2x_1 - 3x_2 \tag{17.39a}$$

$$\text{Subject to:} \quad g_1(\mathbf{x}) = 7 - x_1 - 5x_2 \leq 0 \tag{17.39b}$$

$$g_2(\mathbf{x}) = 10 - 4x_1 - x_2 \leq 0 \tag{17.39c}$$

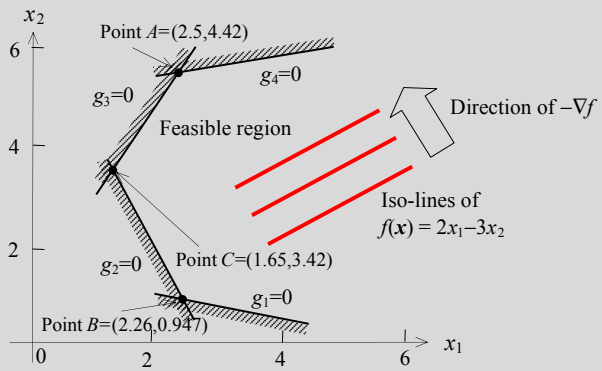
$$g_3(\mathbf{x}) = -7x_1 + 6x_2 - 9 \leq 0 \tag{17.39d}$$

$$g_4(\mathbf{x}) = -x_1 + 6x_2 - 24 \leq 0 \tag{17.39e}$$

Solution

We follow the steps discussed above to illustrate the graphical method.

Step 1: We first sketch the constraint functions g_1 to g_4 on an x_1–x_2 plane shown below.



Continued

EXAMPLE 17.6—cont'd

Step 2: We identify the feasible region to the right of a polyline shown above. Note that the region separated by the four straight lines and to the right of line segments ACB shown above is the feasible region.

Step 3a: We sketch iso-lines of the objective function on top of the feasible region. The negative gradient of the iso-lines is $-\nabla f = [-2, 3]^T$, showing the direction of decreasing objective function $f(\mathbf{x})$. The decreasing iso-line intersects the feasible region at Point A before exiting the region. Therefore, the optimal solution to the LP problem is Point A, in which $f(\mathbf{x}) = f(2.5, 4.42) = -8.26$.

Step 3b: We may calculate the values of the vertices of the feasible region, in this case, $A = (2.5, 4.42)$, $B = (2.26, 0.947)$, and $C = (1.65, 3.42)$. Plugging these values into the objective function, we have $f(A) = -8.26$, $f(B) = 1.68$, and $f(C) = -6.96$. Therefore, the optimal solution is at point A.

This LP problem can be also solved by using the MATLAB function linprog. linprog solves an LP problem in the following form:

$$\min \mathbf{f}^T\mathbf{x} \text{ such that} \begin{cases} \mathbf{A}\cdot\mathbf{x} \leq \mathbf{b}, \\ \mathbf{Aeq}\cdot\mathbf{x} \leq \mathbf{beq}, \\ \mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub}. \end{cases}$$

where **f**, **x**, **beq**, **lb**, and **ub** are vectors; and **A** and **Aeq** are matrices. Entities with "eq" imply equality constraints.

The following script solves the LP problem using MATLAB.

```
f=[2;-3];
A=[-1 -5
-4 -1
-7 6
-1 6];
b=[-7;-10;9;24];
lb=[];
ub=[];
x0=[]; %initial design
[x,fval, exitflag]=linprog(f,A,b,lb,ub,x0); %returns a value exitflag that describes
the exit condition, x:solution, fval: objective function value
```

The solutions returned by MATLAB are

```
>> x, fval
x =
2.5000
4.4167
fval =
-8.2500
```

which are identical to those obtained by using the graphical method.

Note that, in some cases, the solutions obtained may not be unique. For example, in the above problem, if the objective function is redefined as $f(x) = 4x_1 + x_2$, the solution to this problem is all points in line segment between points B and C, as shown in the figure of Example 17.6. The function value is $f(\mathbf{x}) = 4x_1 + x_2 = 10.02$ along line segment BC.

For an LP problem of more than two variables, a method called the simplex method is powerful and widely accepted. For more details regarding the simplex method, please refer to excellent textbooks, such as Arora (2012).

17.4.2 NONLINEAR PROGRAMMING PROBLEMS

When either the objective or any constraint function is nonlinear in terms of the variables, we have a nonlinear programming (NLP) problem. Steps for solving an NLP problem using the graphical method are similar to those of LP problems discussed above, except that graphing these functions may require using software tools, such as MATLAB.

EXAMPLE 17.7

Solve the following NLP optimization problem using the graphical method.

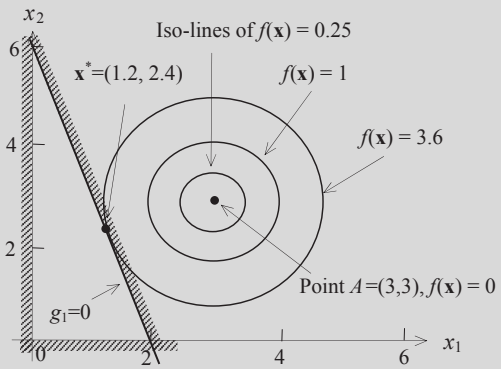
$$\text{Minimize: } f(\mathbf{x}) = (x_1 - 3)^2 + (x_2 - 3)^2 \tag{17.40a}$$

$$\text{Subject to: } g_1(\mathbf{x}) = 3x_1 + x_2 - 6 \leq 0 \tag{17.40b}$$

$$0 \leq x_1, \text{ and } 0 \leq x_2 \tag{17.40c}$$

Solution

Following the same steps discussed above, we sketch the feasible region bound by constraint $g_1(\mathbf{x})$ and side constraints shown below.



We sketch iso-lines of the objective function, which are concentric circles with center at point $A = (3, 3)$. It is clear in the figure above that the optimal point $\mathbf{x}^*$ is a tangent point of $f(\mathbf{x}) = C$ on the straight line $g_1(\mathbf{x}) = 0$. The tangent point can be calculated by intersecting a straight line of slope 1/3 that passes point A; that is, $x_1 - 3x_2 + 6 = 0$, and the straight line $g_1(\mathbf{x}) = 0$. The point is found as $\mathbf{x}^* = (1.2, 2.4)$, in which $f(\mathbf{x}) = 3.6$.

This NLP problem can also be solved by using MATLAB function fmincon, which solves an NLP problem in the following form:

$$\min \mathbf{f}^{\mathrm{T}}\mathbf{x} \text{ such that } \begin{cases} \mathbf{c}(\mathbf{x}) \leq \mathbf{0}, \\ \mathbf{ceq}(\mathbf{x}) = \mathbf{0}, \\ \mathbf{A}\cdot\mathbf{x} \leq \mathbf{b}, \\ \mathbf{Aeq}\cdot\mathbf{x} \leq \mathbf{beq}, \\ \mathbf{lb} \leq \mathbf{x} \leq \mathbf{ub}. \end{cases}$$

Continued

EXAMPLE 17.7—cont'd

The following script solves the NLP problem. We first create the two needed files for objective and constraint functions. We name them, respectively, objfun.m and confun.m. The contents of these two files are shown below. Note that the name of the function needs to match that defined in the main script from the MATLAB window.

```
%contents of the file: objfun.m
function f = objfun(x)
f = (x(1)-3)^2+(x(2)-3)^2;
```

```
%contents of the file: confun.m
function [c, ceq] = confun(x)
% Nonlinear inequality constraints
c = [3*x(1)+x(2)-6 ;-x(1) ;-x(2)] ;
% No nonlinear equality constraints, hence ceq is empty
ceq = [];
```

Enter the following main script:

```
x0 = [0,0]; % Make a starting guess at the solution
Options = optimset('Algorithm','active-set');
% fmincon(objfun,x0,A,b,Aeq,beq,lb,ub,confun, options)
[x,fval] = fmincon(@objfun,x0,[ ],[ ],[ ],[ ],[ ],[ ],@confun, options);
```

The solutions returned by MATLAB are

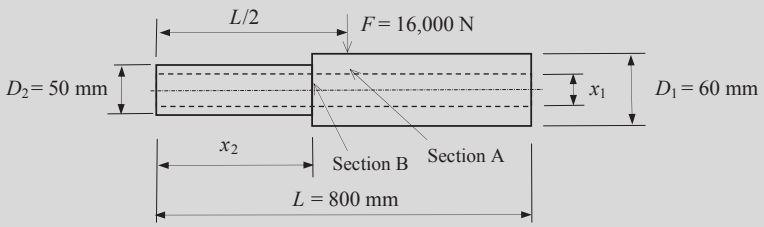
```
>> x, fval
x =
1.2000 2.4000
fval =
3.6000
```

which are identical to those obtained by using the graphical method.

Next, we use a simple beam, a practical engineering problem, to illustrate the graphical method further.

EXAMPLE 17.8

Formulate an optimization problem for the design of a simple support beam of hollow circular cross-section shown below. The beam is loaded with a point force $F = 16{,}000$ N at the mid-section (Section A). The yield strength of the material is 1200 MPa and the required safety factor is $n = 3$. Geometric dimensions of the beam are shown in the figure below. We want to design the beam to minimize its volume subject to the maximum bending stress, no greater than the allowable level of 400 MPa (yield strength divided by safety factor), by varying two design variables: the diameter of the hollow section x_1 and length of the left segment of the beam x_2. The minimum thickness of the beam is 2.5 mm. Diameter x_1 must be no less than 30 mm, and length x_2 must be less than half the beam length. Solve the optimization problem using the graphical method.



EXAMPLE 17.8—cont'd

Solution

We first formulate the optimization problem. The volume of the beam can be calculated as

$$V = \frac{\pi(D_2^2 - x_1^2)}{4}x_2 + \frac{\pi(D_1^2 - x_1^2)}{4}(L - x_2) = \frac{\pi}{4}\left[(2500 - x_1^2)x_2 + (3600 - x_1^2)(800 - x_2)\right]$$

The maximum bending stress is found either at the mid-section (Section A) of the beam where the maximum bending moment occurs or at the junction of the two segments (Section B) where the bending stiffness of the beam is reduced with a relatively large bending moment of a smaller flexural rigidity. The stress measures at these two locations are calculated as

$$\sigma_A = \frac{Mc}{I} = \frac{\left(\frac{FL}{4}\right)\left(\frac{D_1}{2}\right)}{\frac{\pi}{64}(D_1^4 - x_1^4)} = \frac{\left(\frac{16{,}000\times 800}{4}\right)\left(\frac{60}{2}\right)}{\frac{\pi}{64}(60^4 - x_1^4)} = \frac{1.956\times 10^9}{1.296\times 10^7 - x_1^4}$$

$$\sigma_B = \frac{Mc}{I} = \frac{\left(\frac{Fx_2}{2}\right)\left(\frac{D_2}{2}\right)}{\frac{\pi}{64}(D_2^4 - x_1^4)} = \frac{\left(\frac{16{,}000x_2}{2}\right)\left(\frac{50}{2}\right)}{\frac{\pi}{64}(50^4 - x_1^4)} = \frac{4.074\times 10^6 x_2}{6.25\times 10^6 - x_1^4}$$

The optimization problem can then be stated as

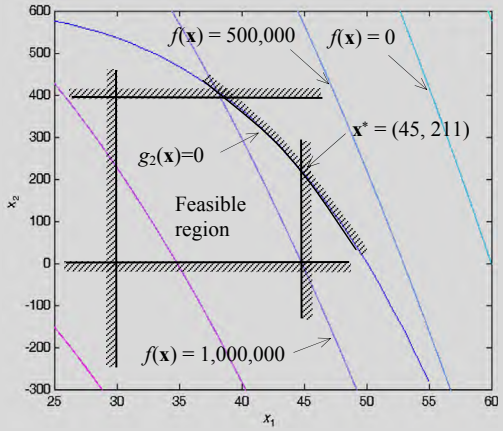
$$\text{Minimize: } f(x_1, x_2) = \frac{\pi}{4}\left[(2500 - x_1^2)x_2 + (3600 - x_1^2)(800 - x_2)\right] \tag{17.41a}$$

$$\text{Subject to: } g_1(x_1, x_2) = \frac{1.956\times 10^9}{1.296\times 10^7 - x_1^4} - 400 \le 0 \tag{17.41b}$$

$$g_2(x_1, x_2) = \frac{4.074\times 10^6 x_2}{6.25\times 10^6 - x_1^4} - 400 \le 0 \tag{17.41c}$$

$$30 \le x_1 \le 45; \quad 0 \le x_2 \le 400 \tag{17.41d}$$

Note that from Eq. 17.41b, we solve for x_1 as $x_1 \le 53.30$ mm. Because the upper bound of the design variable x_1 is 45 mm, the constraint equation $g_1(\mathbf{x}) \le 0$ in Eq. 17.41b is never active. Therefore, it is removed from the optimization problem.



Using the graphical method, we sketch the feasible region bounded by constraint $g_2(\mathbf{x})$ and two side constraints. We also sketch iso-lines of the objective function $f(\mathbf{x})$ shown above. The MATLAB script used for sketching the curves can be found on the book's companion website, http://booksite.elsevier.com/9780123820389 (Script 17.4). As depicted in the graph, optimal point $\mathbf{x}^*$ is identified as the intersection of $g_2(\mathbf{x}) = 0$ and $x_1 = 45$; that is, $\mathbf{x}^* = (45, 211.0)$, in which $f(\mathbf{x}^*) = f(45, 211.0) = 807{,}300\ \text{mm}^3$.

As shown in the examples of this section, the concept of the graphical method is straightforward and the method is easy to use. The only limitation is certainly that the number of design variables cannot be greater than 2. On the other hand, as seen in the examples, one key step in using the graphical method is to sketch the feasible region and graph iso-lines of the objective function in the design space. Therefore, familiarity with graphing software, such as MATLAB, becomes important to use the graphical method for solving optimization problems. In the next sections, we introduce more general approaches, including gradient-based and non-gradient approaches for both constrained and unconstrained problems.

17.5 GRADIENT-BASED APPROACH

The gradient-based approach solves optimization problems by searching in the design space based on the gradients of objective and constraint functions that are active using numerical algorithms. This approach solves for both constrained and unconstrained problems of more than two design variables. However, for illustration purposes, we use examples of one or two design variables.

The gradient-based approach starts with an initial design. This approach searches for a local minimum that is closest to the initial design in an iterative manner. Note that for constrained problems, if the initial design is infeasible, the goal of the search is often to first bring the design into the feasible region. In doing so, the objective function may increase if there is a conflict in design between the objective and constraint functions. A convergent criterion needs to be defined to terminate the search when an optimal solution is found. The criterion must stop the search even if a local minimum is not found after a certain number of design iterations.

In this section, we discuss unconstrained optimization problems using basic search methods. For illustration purposes, we assume problems with only one design variable. As readers quickly figure the limitation of the simple search methods, we introduce more general gradient-based search methods for more design variables. We offer sample MATLAB scripts for solving example problems. MATLAB scripts can be found on the book's companion website, http://booksite.elsevier.com/9780123820389.

17.5.1 GENERATIVE METHOD

A generative method is a brute-force approach, in which the design domain is divided into n equal intervals of Δx between its upper bound x^u and lower bound x^ℓ, and then the objective function $f(x)$ is evaluated at individual designs. For example, at the ith design point (or design), the design variable value is determined by

$$x^i = x^\ell + i\Delta x = x^\ell + \frac{x^u - x^\ell}{n} i \tag{17.42}$$

The function values are evaluated at all points and then compared to find a minimum, as illustrated in Figure 17.10. This method finds global minimum in the design domain. However, the closeness of the found solution to the true optimal depends on the size of the interval Δx. A smaller Δx yields a better result, however, requiring more function evaluations. Some commercial software, such as SolidWorks Simulation, use this method to support design optimization, which is in general inefficient because each function evaluation requires a finite element analysis.

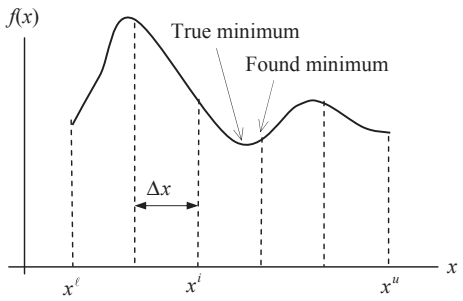


FIGURE 17.10

Illustration of the generative method.

17.5.2 SEARCH METHODS

The simplest search method uses equal intervals in the design variable. As illustrated in Figure 17.11, the minimum of an objective function $f(x)$ of a single variable x is being searched. In general, the explicit expression of the objective function in design variable is not available, in which the objective function is evaluated numerically, for example, using finite element analysis.

In using the equal interval search, the function $f(x)$ is evaluated at points with equal Δx increments. The calculated values of the function at two successive points are then compared. As shown in Figure 17.11, the initial design given is x^0. A search interval Δx is prescribed. If $f(x^0) > f(x^0 + \Delta x)$, then $x^1 = x^0 + \Delta x$ becomes the current design, and the search continues in the positive Δx direction. Otherwise, $x^1 = x^0 - \Delta x$ is the new design, and continue searching in the $-\Delta x$ direction. When an increase in function value at any final point x^f is encountered,

$$f\left(x^f - \Delta x\right) < f\left(x^f\right)$$

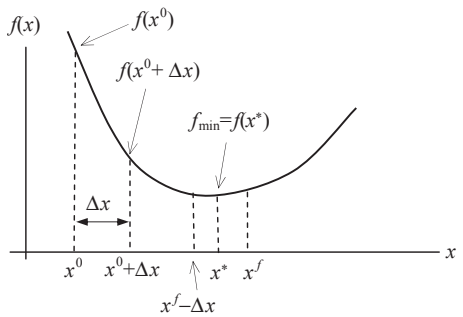


FIGURE 17.11

Illustration of the equal interval search method.

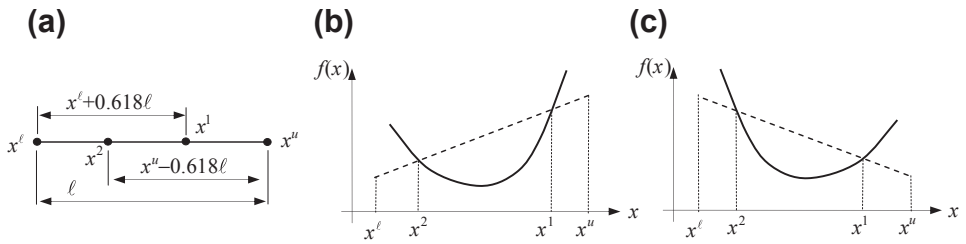


FIGURE 17.12

Illustration of the golden section search method.

the minimum point has passed. At this point, the search direction is reversed and the increment Δx is usually cut in half. Now, the search process is restarted by evaluating the function at $x^f - \Delta x/2$. The process repeats until the following convergent criterion is satisfied:

$$\left|f\left(x^k\right) - f\left(x^{k-1}\right)\right| < \varepsilon \tag{17.43}$$

in which x^k and x^{k-1} represent two consecutive search points, and ε is a prescribed convergence tolerance. This method finds a local minimum that is close to the start point (or initial design) x^0.

An improved version of the search method is called golden section search. In this method, the size of the search interval is changing by a golden ratio 0.618 from the current to the next search. As shown in Figure 17.12(a), the first point is identified as $x^1 = x^\ell + 0.618\ (x^u - x^\ell) = x^\ell + 0.618\ell$, then the second point $x^2 = x^u - 0.618\ell$. The method works as follows:

- If $f(x^1) > f(x^2)$, then the minimum is between x^ℓ and x^1, and the region to the right of x^1 is excluded from the search, as illustrated in Figure 17.12(b). We assign $x^u = x^1$ to continue the search.
- If $f(x^1) < f(x^2)$, then the minimum is between x^2 and x^u, the region to the left of x^2 is excluded from the search, as illustrated in Figure 17.12(c). We assign $x^\ell = x^2$ to continue the search.
- If $|f(x^u) - f(x^\ell)| < \varepsilon$, then we found the optimal point at $x = x^u$ or x^ℓ.

This method offers a better convergent rate than that of the equal interval method, implying that less numbers of function evaluations are needed.

EXAMPLE 17.9

Use the golden search method to find the minimum point of the following function in the interval of $1 \le x \le 11$. We assume the convergent tolerance is $\varepsilon = 0.0001$.

$$f(x) = 3x^3 + 1500/x$$

Solution

From the discussion above, the initial two points can be found as $x^1 = x^\ell + 0.618(x^u - x^\ell) = 1 + 6.18 = 7.18$, and $x^2 = x^u - 0.618\ (x^u - x^\ell) = 11 - 6.18 = 4.82$. Evaluating $f(x)$ at these two points, we have $f(x^1) = 1320$, and $f(x^2) = 647$. Because $f(x^1) > f(x^2)$, we assign $x^u = x^1 = 7.18$ to continue the search.

In the next iteration, we have $x^1 = x^\ell + 0.618\ (x^u - x^\ell) = 1 + 0.618\ (7.18 - 1) = 4.82$, and $x^2 = x^u - 0.618\ (x^u - x^\ell) = 7.18 - 0.618\ (7.18 - 1) = 3.36$. Evaluating $f(x)$ at these two points, we have $f(x^1) = 647$, and $f(x^2) = 560$. Because $f(x^1) > f(x^2)$, we assign $x^u = x^1 = 4.82$ to continue the search. The process repeats until $|f(x^u) - f(x^\ell)| < \varepsilon = 0.0001$.

The golden search method can be implemented in MATLAB for solving this example, such as script 17.5 on the book's companion website, http://booksite.elsevier.com/9780123820389. The optimal point is found at $x^* = 3.591$, in which $f(x^*) = 556.6$.

These basic search methods are simple and easy to implement. However, they take a lot of function evaluations to find an optimal solution. When the number of design variables increases, the computation time required for performing function evaluations is substantial, making these methods less desirable.

17.5.3 GRADIENT-BASED SEARCH

Another kind of search method is guided by the gradient of the objective function. The concept of the gradient-based search is illustrated in Figure 17.13. As shown in Figure 17.13, an initial design is given as x^0. The derivative (or gradient) of the objective function $f(x)$ is calculated at x^0 as $f'(x^0)$, which is the slope of the objective function curve shown in Figure 17.13. The gradient of the objective function $f'(x^0)$, which is the slope of the objective function, determines the search direction—in this case, in the direction of decreasing the objective function. Once the search direction is determined, the step size along the direction is sought. Usually, a relatively large step size Δx can be assumed to find the variable value of the next iteration; that is, $x^1 = x^0 + \Delta x$. Once x^1 is determined, the objective function value $f(x^1)$ and gradient $f'(x^1)$ are calculated. Usually, the same step size Δx is employed to determine the next design until the search direction is reversed, for example, at x^2, as shown in Figure 17.13. At this point, the step size is reduced to $\Delta x/2$ and the search is resumed. This is called the bisection search. Determining an adequate step size along the search direction is called a line search. There are several prominent algorithms supporting line search in optimization, such as the conjugate gradient method, backtracking line search, or Wolfe conditions.

The process repeats until a convergence criterion is met. Note that the convergence criteria can be defined in different ways. For example, the difference in the objective function values of two consecutive iterations is less than a prescribed convergent tolerance ε_1:

$$\left| f\left(x^k\right) - f\left(x^{k-1}\right) \right| < \varepsilon_1 \tag{17.44a}$$

Or, the magnitude of the gradient of the objective function is less than a prescribed convergent tolerance ε_2:

$$\left| f'\left(x^k\right) \right| < \varepsilon_2 \tag{17.44b}$$

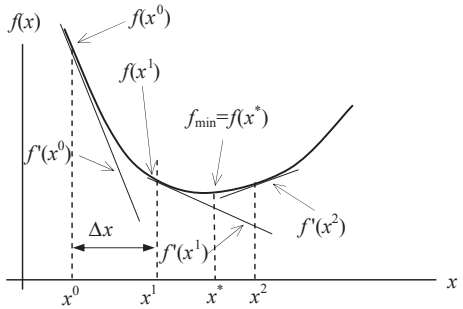


FIGURE 17.13

Illustration of the gradient-based search method.

EXAMPLE 17.10

Use the gradient-based method to find the minimum point of the same objective function as that of Example 17.9 in the interval of $1 \leq x \leq 11$. We assume the initial design is given at $x^0 = 1$ and convergent tolerance is $\varepsilon_1 = 0.0001$.

Solution

First, the derivative of the objective function, $f(x) = 3x^3 + 1500/x$, is

$$f'(x) = 9x^2 - 1500/x^2$$

At the initial design, $f(x^0) = f(1) = 1503$, and $f'(x^0) = f'(1) = -1411$. We assume the initial step size is $\Delta x = (x^u - x^\ell)/2 = (11 - 1)/2 = 5$. Hence, the next design variable is $x^1 = x^0 + \Delta x = 1 + 5 = 6$.

At the new design, we calculate the function value and its gradient as $f(x^1) = f(6) = 898$, and $f'(x^1) = f'(6) = 282$, which is reversed from the previous design point. Hence, the step size is reduced to half, $\Delta x = 5/2 = 2.5$; and the next design point is $x^2 = x^1 - \Delta x = 6 - 2.5 = 3.5$. The process repeats until $|f(x^i) - f(x^{i-1})| < \varepsilon_1 = 0.0001$.

The gradient-based search method can be implemented in MATLAB to solve the example (Script 17.6). The optimal point is found at $x^* = 3.594$, in which $f(x^*) = 556.6$.

Next, we considered objective functions of multiple design variables. In general, the two major factors to determine in gradient-based search methods are search direction and line search. For unconstrained problems, the search direction is determined by the gradient vector of the objective function. Once the search direction is determined, an appropriate step size along the search direction must be determined to locate the next design point, which is called line search. Although several methods can be used to find the optimal solution of multivariable objective functions, the steepest descent method, which is one of the simplest methods, as well as an improvement from the concept of steepest descent called the conjugate gradient method, are discussed. In this subsection, we assume the bisection method discussed just now for line search. More about line search is provided in Section 17.5.4.

17.5.3.1 Steepest Descent Method

The gradient-based methods rely on the gradient vector to determine the search direction in finding an optimal solution. For an objective function of n variables $f(\mathbf{x})$, the gradient vector is defined as

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_n}\right]^{\mathrm{T}} \tag{17.45}$$

As discussed in Example 17.2, the gradient vector at a point $\mathbf{x}$ defines the direction of maximum increase in the objective function. Thus, the direction of maximum decrease is opposite to that—that is, negative of the gradient vector $-\nabla f(\mathbf{x})$. Any small move in the negative gradient direction will result in the maximum local rate of decrease in the objective function. The negative gradient vector thus represents a direction of steepest descent for the objective function and is written as

$$\mathbf{n} = -\nabla f(\mathbf{x}) = -\left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_n}\right]^{\mathrm{T}} \tag{17.46}$$

or

$$n_i = -\frac{\partial f}{\partial x_i}, \quad i = 1, n \tag{17.47}$$

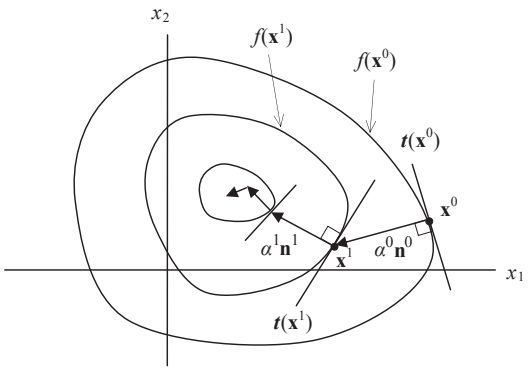


FIGURE 17.14

Illustration of the orthogonal steepest descent path.

In general, the method of steepest descent, also called the gradient descent method, starts with an initial point $\mathbf{x}^0$ and, as many times as needed, moves from $\mathbf{x}^k$ to $\mathbf{x}^{k+1}$ by minimizing along the vector $\mathbf{n}^k$ extending from $\mathbf{x}^k$ in the direction of $-\nabla f(\mathbf{x}^k)$, the local downhill gradient. The vector $\mathbf{n}^k$, representing the search direction for minimizing the objective function along the steepest descent direction, is usually normalized as

$$\mathbf{n}^k = \mathbf{n}\left(\mathbf{x}^k\right) = \frac{-\nabla f\left(\mathbf{x}^k\right)}{\left\|\nabla f\left(\mathbf{x}^k\right)\right\|} \tag{17.48}$$

Notice the same notation $\mathbf{n}$ is employed for the normalized search direction.

As mentioned before, the steepest descent direction at a point $\mathbf{x}^k$ determines the direction of maximum decrease in the objective function. Once the search direction is calculated, line search is carried out to find a step size α^k along the search direction. The next design is then determined as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{n}^k \tag{17.49}$$

If we use the bisection search method, the step size α is reduced by half if the function value at the current design is greater than the previous one. The process repeats until the difference in the objective function values of two consecutive iterations is less than a prescribed convergent tolerance ε_1, or the magnitude of the gradient is less than a prescribed tolerance; that is, $||\nabla f(\mathbf{x}^k)|| < \varepsilon_2$. Note that geometrically, the steepest descent direction of a design $\mathbf{x}^k$ is perpendicular to the tangent line $\mathbf{t}(\mathbf{x}^k)$ of the iso-line of the objective function $f(\mathbf{x}^k)$, as shown in Figure 17.14. Hence, the search path is a zigzag polyline shown in Figure 17.14.

EXAMPLE 17.11

Use the steepest descent method to find the minimum point of the objective function

$$f(x_1, x_2) = \frac{(x_1 - 2)^2}{4} + (x_2 - 1)^2$$

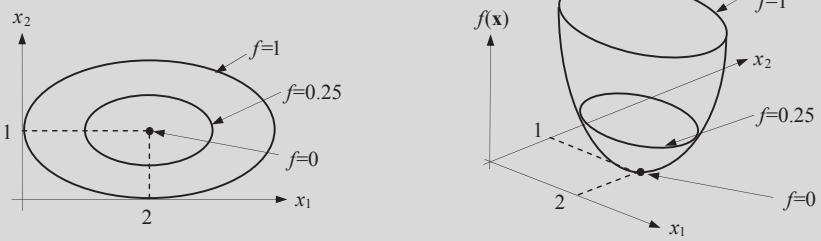
We assume the initial design is given at $\mathbf{x}^0 = (0, 0)$, the step size is $\alpha = 2$, and convergent tolerance is $\varepsilon_1 = 0.00001$.

Continued

EXAMPLE 17.11—cont'd

Solution

We first plot the function for better illustration. The iso-lines of the objective function are ellipse with center point at (2, 1), and with a ratio of long and short axes 2:1. Two iso-lines of $f = 0.25$ and $f = 1$ are shown in the figure below (left). The elliptic cone shown in the figure below (right) depicts the objective function viewed in three-dimensional (3-D) space. Two ellipses corresponding to $f = 0.25$ and $f = 1$ are shown in the figure. The minimum is found at the center of the ellipses; that is, $(x_1, x_2) = (2, 1)$.



The gradient of the objective function can be calculated as

$$\nabla f(x_1, x_2) = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2}\right]^{\mathrm{T}} = \left[\frac{1}{2}(x_1 - 2) \quad 2(x_2 - 1)\right]^{\mathrm{T}}$$

As before, we start with an initial design, which is assumed as $\mathbf{x}^0 = (0, 0)$. At this point, the function value and gradient are $f(\mathbf{x}^0) = f(0, 0) = 2$ and $\nabla f(\mathbf{x}^0) = [-1, -2]^{\mathrm{T}}$, respectively. Also, the search direction is calculated using Eq. 17.48 as $\mathbf{n}^0 = \mathbf{n}(\mathbf{x}^0) = [1/\sqrt{5}, 2/\sqrt{5}]^{\mathrm{T}}$.

We assume $\alpha^0 = 2$, and we use the bisection search method as before. Therefore, from Eq. 17.49 we calculate the next point $\mathbf{x}^1$ as

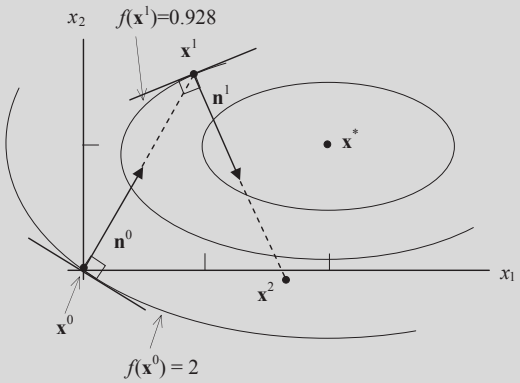
$$\mathbf{x}^1 = \mathbf{x}^0 + \alpha^0\mathbf{n}^0 = [0, 0]^{\mathrm{T}} + 2\left[1/\sqrt{5}, 2/\sqrt{5}\right]^{\mathrm{T}} = \left[2/\sqrt{5}, 2/\sqrt{5}\right]^{\mathrm{T}} = [0.894, 1.79]^{\mathrm{T}}$$

At the new point, the function value, gradient, and search direction are $f(\mathbf{x}^1) = f(0.894, 1.79) = 0.928$, $\nabla f(\mathbf{x}^1) = [-0.553, 1.58]^{\mathrm{T}}$, and $\mathbf{n}^1 = [0.330, -0.944]^{\mathrm{T}}$, respectively. Then, we calculate the next point $\mathbf{x}^2$ as

$$\mathbf{x}^2 = \mathbf{x}^1 + \alpha^1\mathbf{n}^1 = [0.894, 1.79]^{\mathrm{T}} + 2[0.330, -0.944]^{\mathrm{T}} = [1.56, -0.0986]^{\mathrm{T}}$$

At the new point, the function value is $f(\mathbf{x}^2) = f(1.56, -0.0986) = 1.42 > f(\mathbf{x}^1)$. Hence, step size α is reduced to $\alpha^2 = \alpha^0/2$. The process continues until $|f(\mathbf{x}^k) - f(\mathbf{x}^{k-1})| < \varepsilon_1 = 0.00001$.

The gradient-based search method can be implemented in MATLAB for solving the example (see Script 17.6). The optimal point is found at $\mathbf{x}^* = (2.0003, 1.0018)$, in which $f(\mathbf{x}^*) = 4.49 \times 10^{-6}$. The first few iterations of the search path are illustrated in the figure below.



17.5.3.2 Conjugate Gradient Method

The conjugate gradient method is a simple and effective modification of the steepest descent method. We start with an initial design $\mathbf{x}^0$, set the convergence tolerance ε, and calculate the function value $f(\mathbf{x}^0)$ and gradient vector $\nabla f(\mathbf{x}^0)$. With the gradient vector calculated, we calculate the search direction $\mathbf{n}^0$ using Eq. 17.48. After this first iteration, instead of continuously using Eq. 17.48 to calculate the search direction as in the steepest descent method, the conjugate gradient method searches optimal design along the conjugate direction defined by

$$\mathbf{n}^k = \frac{-\nabla f(\mathbf{x}^k)}{\|\nabla f(\mathbf{x}^k)\|} + \beta^k \mathbf{n}^{k-1} \tag{17.50}$$

in which the parameter β^k is defined in several different ways, for example, by Fletcher and Reeves (1964) as

$$\beta^k = \frac{\|\nabla f(\mathbf{x}^k)\|}{\|\nabla f(\mathbf{x}^{k-1})\|} \tag{17.51}$$

After the first iteration, the only difference between the conjugate gradient and steepest descent methods is in Eq. 17.50. In this equation, the current steepest descent direction is modified by adding a scaled direction that was used in the previous iteration. The scale factor β^k is determined by using lengths of the gradient vector at the two iterations, as shown in Eq. 17.51. Thus, the conjugate direction is nothing but a deflected steepest descent direction. This is a simple modification that requires little additional calculation. In general, this method improves the rate of convergence of the steepest descent method.

EXAMPLE 17.12

Use the conjugate gradient method to find the minimum point of the same function of Example 17.11 with a convergent tolerance $\varepsilon_1 = 0.00001$.

Solutions

The first iteration is identical to the steepest descent method; that is, $\nabla f(\mathbf{x}^0) = [-1, -2]^T$, $\mathbf{n}^0 = \mathbf{n}(\mathbf{x}^0) = [1/\sqrt{5}, 2/\sqrt{5}]^T$, $\mathbf{x}^1 = [0.894, 1.79]^T$, $\nabla f(\mathbf{x}^1) = [-0.553, 1.58]^T$, and $\mathbf{n}^1 = [0.330, -0.944]^T$. From Eq. 17.51:

$$\beta^1 = \frac{\|\nabla f(\mathbf{x}^1)\|}{\|\nabla f(\mathbf{x}^0)\|} = \frac{\|[-0.533, 1.58]^T\|}{\|[-1, -2]^T\|} = \frac{1.67}{2.24} = 0.746$$

Hence, from Eq. 17.50, the search direction becomes

$$\mathbf{n}^1 = -\frac{\nabla f(\mathbf{x}^1)}{\|\nabla f(\mathbf{x}^1)\|} + \beta^1 \mathbf{n}^0 = -\frac{[-0.533, 1.58]^T}{1.67} + 0.764\left[1/\sqrt{5}, 2/\sqrt{5}\right]^T = [0.661, -0.263]^T$$

The vector $\mathbf{n}^1$ is normalized as $\mathbf{n}^1 = [0.929, -0.370]^T$. Then, we calculate the next point $\mathbf{x}^2$ as

$$\mathbf{x}^2 = \mathbf{x}^1 + \alpha^1 \mathbf{n}^1 = [0.894, 1.79]^T + 2[0.929, -0.370]^T = [2.75, 1.05]^T$$

At the new point, the function value is $f(\mathbf{x}^2) = f(2.75, 1.05) = 0.143 < f(\mathbf{x}^1)$. Hence, the process continues with the same step size α. The process continues until $|f(\mathbf{x}^k) - f(\mathbf{x}^{k-1})| < \varepsilon_1 = 0.00001$.

The conjugate gradient method can be implemented in MATLAB to solve the example (see Script 17.8 for details). The optimal point is found at $\mathbf{x}^* = (1.9680, 1.0133)$, in which $f(\mathbf{x}^*) = 4.34 \times 10^{-4}$.

17.5.3.3 Quasi-Newton Method

With the steepest descent method, only first-order derivative information is used to determine the search direction. If second-order derivatives are available, we can use them to represent the objective function more accurately, and a better search direction can be found. With the inclusion of second-order information, we can expect a better rate of convergence as well. For example, Newton's method, which uses the Hessian of the function in calculating the search direction, has a quadratic rate of convergence (meaning that it converges very rapidly when the design point is within a certain radius of the minimum point).

The basic idea of the classical Newton's method is to use a second-order Taylor's series expansion of the function around the current design point $\mathbf{x}$, a vector of size of $n \times 1$ (n is the number of design variables):

$$f(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x}) + \nabla f^{\mathrm{T}}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^{\mathrm{T}}\mathbf{H}\Delta\mathbf{x} \tag{17.52}$$

where $\Delta\mathbf{x}$ is a small change in design and $\mathbf{H}_{n\times n}$ is the Hessian of the objective function f at the point $\mathbf{x}$. Employing the optimality condition to Eq. 17.52 by taking derivative of Eq. 17.52 with respect to $\Delta\mathbf{x}$ and setting the derivative to zero—that is, $\partial f/\partial(\Delta\mathbf{x}) = 0$—we have

$$\nabla f + \mathbf{H}\Delta\mathbf{x} = 0 \tag{17.53}$$

Assuming $\mathbf{H}$ to be nonsingular, we get an expression for $\Delta\mathbf{x}$ as

$$\Delta\mathbf{x} = -\mathbf{H}^{-1}\nabla f \tag{17.54}$$

where $\Delta\mathbf{x}$ updates design to the next point; that is, $\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}$. Because Eq. 17.52 is just an approximation for f at the current point $\mathbf{x}^k$, the next design point $\mathbf{x}^{k+1}$ updated using $\Delta\mathbf{x}$ will most likely not be the precise minimum point of $f(\mathbf{x})$. Therefore, the process will have to be repeated to obtain improved estimates until a minimum is reached.

Newton's method is inefficient because it requires calculation of $n(n+1)/2$ second-order derivatives to generate the Hessian matrix. For most engineering design problems, calculation of second-order derivatives may be too expensive to perform or entirely infeasible numerically due to poor accuracy. Also, Newton's method runs into difficulties if the Hessian of the function is singular at any iteration.

Several methods were proposed to overcome the drawbacks of Newton's method by generating an approximation for the Hessian matrix or its inverse at each iteration. Only the first derivatives of the function are used to generate these approximations. They are called quasi-Newton methods. We introduce the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method, which updates the Hessian rather than its inverse at every iteration.

17.5.3.4 The BFGS Method

The search direction $\mathbf{n}^k$ is given by the solution of the Newton equation in Eq. 17.54:

$$\mathbf{H}^k\mathbf{n}^k = -\nabla f\left(\mathbf{x}^k\right) \tag{17.55}$$

where $\mathbf{H}^k = \mathbf{H}(\mathbf{x}^k)$ is an approximation to the Hessian matrix which is updated iteratively at each design iteration. Note that at the initial design, the Hessian matrix is set to be an identity matrix; that is, $\mathbf{H}^0 = \mathbf{I}$. A line search in the direction $\mathbf{n}^k$ is then carried out to find the next point $\mathbf{x}^{k+1}$, as shown in Eq. 17.49.

The approximate Hessian at the design iteration $k+1$ is updated by the addition of two matrices:

$$\mathbf{H}^{k+1} = \mathbf{H}^k + \mathbf{D}^k + \mathbf{E}^k \tag{17.56}$$

where the correction matrices $\mathbf{D}^k$ and $\mathbf{E}^k$ are given as

$$\mathbf{D}^k = \frac{\mathbf{y}^k \left(\mathbf{y}^k\right)^{\mathrm{T}}}{\left(\mathbf{y}^k\right)^{\mathrm{T}} \cdot \mathbf{s}^k} \tag{17.57}$$

and

$$\mathbf{E}^k = \frac{\nabla f^k \left(\nabla f^k\right)^{\mathrm{T}}}{\left(\nabla f^k\right)^{\mathrm{T}} \cdot \mathbf{n}^k} \tag{17.58}$$

where $\nabla f^k = \nabla f(\mathbf{x}^k) = [\partial f(\mathbf{x}^k)/\partial x_1, \ldots, \partial f(\mathbf{x}^k)/\partial x_n]^{\mathrm{T}}$, $\mathbf{y}^k = \nabla f^{k+1} - \nabla f^k$, and $\mathbf{s}^k = \alpha^k \mathbf{n}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$.

EXAMPLE 17.13

Use the BFGS method to find the minimum point of the same objective function of Example 17.11 with a convergent tolerance $\varepsilon_2 = 0.001$. In this example, we assume the initial step size to be $\alpha = 1$.

Solution

The first iteration is identical to the steepest descent method; that is, $\nabla f(\mathbf{x}^0) = [-1, -2]^{\mathrm{T}}$, $\mathbf{n}^0 = [1/\sqrt{5}, 2/\sqrt{5}]^{\mathrm{T}}$, $\mathbf{x}^1 = \mathbf{x}^0 + \alpha \mathbf{n}^0 = [0.0]^{\mathrm{T}} + [1/\sqrt{5}, 2/\sqrt{5}]^{\mathrm{T}} = [0.447, 0.894]^{\mathrm{T}}$, and $\nabla f(\mathbf{x}^1) = [-0.776, -0.211]^{\mathrm{T}}$. Using the BFGS method, we first create a Hessian matrix at the initial design as an identity matrix $\mathbf{H}^0 = \mathbf{I}_{2\times 2}$. Then, we update the Hessian matrix at $\mathbf{x}^1$ by calculating matrices $\mathbf{D}^0$ and $\mathbf{E}^0$. For the matrix $\mathbf{D}^0$, we first calculate $\mathbf{y}^0$ and $\mathbf{s}^0$ as

$$\mathbf{y}^0 = \nabla f\left(\mathbf{x}^1\right) - \nabla f\left(\mathbf{x}^0\right) = [-0.776, -0.211]^{\mathrm{T}} - [-1, -2]^{\mathrm{T}} = [0.224, 1.79]^{\mathrm{T}}$$

and

$$\mathbf{s}^0 = \alpha^0 \mathbf{n}^0 = [0.447, 0.894]^{\mathrm{T}} - [0, 0]^{\mathrm{T}} = [0.447, 0.894]^{\mathrm{T}}$$

Hence, from Eq. 17.57, we have

$$\mathbf{D}^0 = \frac{\mathbf{y}^0 \left(\mathbf{y}^0\right)^{\mathrm{T}}}{\left(\mathbf{y}^0\right)^{\mathrm{T}} \cdot \mathbf{s}^0} = \frac{\begin{bmatrix} 0.224 \\ 1.79 \end{bmatrix} [0.224 \quad 1.79]}{[0.224 \quad 1.79] \begin{bmatrix} 0.447 \\ 0.894 \end{bmatrix}} = \frac{\begin{bmatrix} 0.0502 & 0.401 \\ 0.401 & 3.20 \end{bmatrix}}{1.70} = \begin{bmatrix} 0.0295 & 0.236 \\ 0.236 & 1.88 \end{bmatrix}$$

Then, from Eq. 17.58, we have

$$\mathbf{E}^0 = \frac{\nabla f^0 \left(\nabla f^0\right)^{\mathrm{T}}}{\left(\nabla f^0\right)^{\mathrm{T}} \cdot \mathbf{n}^0} = \frac{\begin{bmatrix} -1 \\ -2 \end{bmatrix} [-1 \quad -2]}{[-1 \quad -2] \begin{bmatrix} 1/\sqrt{5} \\ 2/\sqrt{5} \end{bmatrix}} = \frac{\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}}{-\sqrt{5}} = \begin{bmatrix} -0.447 & -0.894 \\ -0.894 & -1.79 \end{bmatrix}$$

The Hessian matrix is now updated according to Eq. 17.56 as

$$\mathbf{H}^1 = \mathbf{H}^0 + \mathbf{D}^0 + \mathbf{E}^0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0.0295 & 0.236 \\ 0.236 & 1.88 \end{bmatrix} + \begin{bmatrix} -0.447 & -0.894 \\ -0.898 & -1.79 \end{bmatrix} = \begin{bmatrix} 0.583 & -0.658 \\ -0.659 & 1.09 \end{bmatrix}$$

Continued

EXAMPLE 17.13—cont'd

Hence, from Eq. 17.55, we have

$$\mathbf{n}^1 = -\left(\mathbf{H}^1\right)^{-1}\nabla f\left(\mathbf{x}^1\right) = -\begin{bmatrix} 0.582 & -0.658 \\ -0.658 & 1.09 \end{bmatrix}^{-1}\begin{bmatrix} -0.776 \\ -0.211 \end{bmatrix} = \begin{bmatrix} 4.89 \\ 3.14 \end{bmatrix}$$

The vector $\mathbf{n}^1$ is normalized as $\mathbf{n}^1 = [0.841, 0.540]^{\mathrm{T}}$. Then, we calculate the next point $\mathbf{x}^2$ as

$$\mathbf{x}^2 = \mathbf{x}^1 + \alpha^1\mathbf{n}^1 = [0.447, 0.894]^{\mathrm{T}} + 1[0.841, 0.540]^{\mathrm{T}} = [1.29, 1.43]^{\mathrm{T}}$$

At the new point, the function value is $f(\mathbf{x}^2) = f(1.29, 1.43) = 0.316 > f(\mathbf{x}^1)$. Hence, step size α is unchanged. The process continues until $||\nabla f(\mathbf{x}^k)|| < \varepsilon_2 = 0.001$.

The gradient-based search method can be implemented in MATLAB for solving the example (see Script 11.9 for details). The optimal point is found at $\mathbf{x}^* = (2, 1)$, in which $f(\mathbf{x}^*) = 0$.

17.5.4 LINE SEARCH

As mentioned earlier, the purpose of the line search is to determine an appropriate step size α along the search direction $\mathbf{n}$ in searching for an optimal solution. Up to this point in our discussion, we employed interval-reducing methods, such as the bisection method, for line search in our discussion and examples. These methods are simple but can require many function evaluations (and many design iterations) to reach an optimum. In engineering design problems, function evaluation requires a significant amount of computational effort. Therefore, these methods may not be desired for practical applications. Because line search is an important step in optimization, we offer a few more details on this subject. We will go over two popular methods: secant method and quadratic curve fitting.

17.5.4.1 Concept of Line Search

As discussed in Section 17.5.3, when a search direction $\mathbf{n}^k$ is found, the next design point $\mathbf{x}^{k+1}$ is determined by a step size α using Eq. 17.49. At the new design, we expect to have a reduced objective function value:

$$f\left(\mathbf{x}^{k+1}\right) = f\left(\mathbf{x}^k + \alpha\mathbf{n}^k\right) \leq f\left(\mathbf{x}^k\right) \tag{17.59}$$

A step size α that reduces the objective function the most is desirable. Therefore, a line search problem can be formulated as a subproblem:

$$\underset{\alpha \geq 0}{\text{Minimize}} \quad f(\alpha) = f\left(\mathbf{x}^k + \alpha\mathbf{n}^k\right) \tag{17.60}$$

Because $\mathbf{x}^k$ and $\mathbf{n}^k$ are known, this problem reduces to a minimization problem of a single variable α. Assuming that $f(\mathbf{x})$ is smooth and continuous, we find its optimum where its first-derivative is set to zero:

$$f'(\alpha) = \frac{\mathrm{d}f\left(\mathbf{x}^k + \alpha\mathbf{n}^k\right)}{\mathrm{d}\alpha} = f\left(\mathbf{x}^k\right) + \nabla f\left(\mathbf{x}^k\right)^{\mathrm{T}}\left(\alpha\mathbf{n}^k\right) = 0 \tag{17.61}$$

The optimization problem of Eq. 17.60 is converted into a root-finding problem of Eq. 17.61, in which the step size α is sought. Mathematically, the step size α can be found as

$$\alpha = -\frac{f\left(\mathbf{x}^k\right)}{\nabla f\left(\mathbf{x}^k\right)^{\mathrm{T}}\mathbf{n}^k} \tag{17.62}$$

in which all the quantities on the right-hand side are known at iteration k. This is nothing but Newton's method.

EXAMPLE 17.14

Use the BFGS method combined with the Newton's method for line search to find the minimum point of the same objective function of Example 17.11 with a convergent tolerance $\varepsilon_2 = 0.001$.

Solution

The first iteration is identical to the steepest descent method; that is, $\nabla f(\mathbf{x}^0) = [-1, -2]^{\mathrm{T}}$, $\mathbf{n}^0 = [1/\sqrt{5}, 2/\sqrt{5}]^{\mathrm{T}}$. From Eq. 17.62:

$$\alpha^0 = -\frac{f\left(\mathbf{x}^0\right)}{\nabla f(\mathbf{x}^0)^{\mathrm{T}}\mathbf{n}^0} = \frac{-2}{\begin{bmatrix}-1 & -2\end{bmatrix}\begin{bmatrix}1/\sqrt{5}\\ 2/\sqrt{5}\end{bmatrix}} = 0.894$$

Then, $\mathbf{x}^1 = \mathbf{x}^0 + \alpha\mathbf{n}^0 = [0,0]^{\mathrm{T}} + 0.894\,[1/\sqrt{5}, 2/\sqrt{5}]^{\mathrm{T}} = [0.400, 0.800]^{\mathrm{T}}$, and $\nabla f(\mathbf{x}^1) = [-0.800, -0.400]^{\mathrm{T}}$. Using the BFGS method, we first create a Hessian matrix at the initial design as an identity matrix $\mathbf{H}^0 = \mathbf{I}_{2\times 2}$. Then we update the Hessian matrix at $\mathbf{x}^1$ by calculating matrices $\mathbf{D}^0$ and $\mathbf{E}^0$. For the matrix $\mathbf{D}^0$ we first calculate $\mathbf{y}^0$ and $\mathbf{s}^0$ as

$$\mathbf{y}^0 = \nabla f\left(\mathbf{x}^1\right) - \nabla f\left(\mathbf{x}^0\right) = [-0.800, -0.400]^{\mathrm{T}} - [-1, -2]^{\mathrm{T}} = [0.200, 1.60]^{\mathrm{T}}$$

and

$$\mathbf{s}^0 = \alpha^0\mathbf{n}^0 = 0.894[0.447, 0.894]^{\mathrm{T}} - [0, 0]^{\mathrm{T}} = [0.400, 0.800]^{\mathrm{T}}$$

Hence from Eq. 17.57, we have

$$\mathbf{D}^0 = \frac{\mathbf{y}^0(\mathbf{y}^0)^{\mathrm{T}}}{(\mathbf{y}^0)^{\mathrm{T}}\cdot\mathbf{s}^0} = \frac{\begin{bmatrix}0.200\\ 1.60\end{bmatrix}\begin{bmatrix}0.200 & 1.60\end{bmatrix}}{\begin{bmatrix}0.200 & 1.60\end{bmatrix}\begin{bmatrix}0.400\\ 0.800\end{bmatrix}} = \frac{\begin{bmatrix}0.0400 & 0.320\\ 0.320 & 2.56\end{bmatrix}}{1.36} = \begin{bmatrix}0.0294 & 0.235\\ 0.235 & 1.88\end{bmatrix}$$

Then, from Eq. 17.58, we have

$$\mathbf{E}^0 = \frac{\nabla f^0(\nabla f^0)^{\mathrm{T}}}{(\nabla f^0)^{\mathrm{T}}\cdot\mathbf{n}^0} = \frac{\begin{bmatrix}-1\\ -2\end{bmatrix}\begin{bmatrix}-1 & -2\end{bmatrix}}{\begin{bmatrix}-1 & -2\end{bmatrix}\begin{bmatrix}1/\sqrt{5}\\ 2/\sqrt{5}\end{bmatrix}} = \frac{\begin{bmatrix}1 & 2\\ 2 & 4\end{bmatrix}}{-\sqrt{5}} = \begin{bmatrix}-0.447 & -0.894\\ -0.894 & -1.79\end{bmatrix}$$

The Hessian matrix is now updated according to Eq. 17.56 as

$$\mathbf{H}^1 = \mathbf{H}^0 + \mathbf{D}^0 + \mathbf{E}^0 = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + \begin{bmatrix}0.0295 & 0.236\\ 0.236 & 1.88\end{bmatrix} + \begin{bmatrix}-0.447 & -0.894\\ -0.898 & -1.79\end{bmatrix} = \begin{bmatrix}0.583 & -0.658\\ -0.659 & 1.09\end{bmatrix}$$

Hence, from Eq. 17.55, we have

$$\mathbf{n}^1 = -\left(\mathbf{H}^1\right)^{-1}\nabla f\left(\mathbf{x}^1\right) = -\begin{bmatrix}0.582 & -0.658\\ -0.658 & 1.09\end{bmatrix}^{-1}\begin{bmatrix}-0.800\\ -0.400\end{bmatrix} = \begin{bmatrix}5.63\\ 3.76\end{bmatrix}$$

Continued

EXAMPLE 17.14—cont'd

The vector $\mathbf{n}^1$ is normalized as $\mathbf{n}^1 = [0.832, 0.555]^T$. Then, we calculate the step size:

$$\alpha^1 = -\frac{f(\mathbf{x}^1)}{\nabla f(\mathbf{x}^1)^T \mathbf{n}^1} = -\frac{0.680}{[-0.800 \quad -0.400]\begin{bmatrix}0.832\\0.555\end{bmatrix}} = 0.766$$

Therefore, the next point $\mathbf{x}^2$ is

$$\mathbf{x}^2 = \mathbf{x}^1 + \alpha^1 \mathbf{n}^1 = [0.400, 0.800]^T + 0.766\,[0.832, 0.555]^T = [1.04, 1.23]^T$$

At the new point, the function value is $f(\mathbf{x}^2) = f(1.04, 1.23) = 0.283$. The process continues until $||\nabla f(\mathbf{x}^k)|| < \varepsilon_2 = 0.001$.

The gradient-based search method can be implemented in MATLAB for solving the example (see Script 17.10 for details). The optimal point is found at $\mathbf{x}^* = (2, 1)$, in which $f(\mathbf{x}^*) = 0$.

17.5.4.2 Secant Method

Because, in practical applications, the gradient vector is expensive to calculate, the secant method is often employed to approximate the gradient vector. To simplify the notation, we assume a function of a single variable for the time being.

We start with two estimates of the design points, x^0 and x^1. The iterative formula, for $k \geq 1$, is

$$x^{k+1} = x^k - \frac{f(x^k)}{q(x^{k-1}, x^k)} \quad \text{or} \quad \Delta x = x^{k+1} - x^k = -\frac{f(x^k)}{q(x^{k-1}, x^k)} \tag{17.63}$$

where

$$q(x^{k-1}, x^k) = \frac{f(x^{k-1}) - f(x^k)}{x^{k-1} - x^k} \tag{17.64}$$

Note that Eq. 17.63 is nothing but Newton's method, except that the denominator in the fraction is replaced by $q(x^{k-1}, x^k)$ defined in Eq. 17.64, which approximates the gradient of the objective function. Note that if x^k is close to x^{k-1}, then $q(x^{k-1}, x^k)$ is close to $f'(x^k)$, and the secant method and Newton's method are virtually identical.

For a function of multivariables, Eqs 17.63 and 17.64 can be written as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{n}^k = \mathbf{x}^k - \frac{f(\mathbf{x}^k)}{\mathbf{q}(\mathbf{x}^{k-1}, \mathbf{x}^k)^T \mathbf{n}^k}\mathbf{n}^k \tag{17.65a}$$

or

$$\alpha^k = -\frac{f(\mathbf{x}^k)}{\mathbf{q}(\mathbf{x}^{k-1}, \mathbf{x}^k)^T \mathbf{n}^k} \tag{17.65b}$$

Note that in Eq. 17.65b, we have

$$\mathbf{q}(\mathbf{x}^{k-1}, \mathbf{x}^k) = \left[\frac{f(\mathbf{x}^{k-1}) - f(\mathbf{x}_1^k)}{x_1^{k-1} - x_1^k} \quad \frac{f(\mathbf{x}^{k-1}) - f(\mathbf{x}_2^k)}{x_2^{k-1} - x_2^k} \cdots \frac{f(\mathbf{x}^{k-1}) - f(\mathbf{x}_n^k)}{x_n^{k-1} - x_n^k}\right]^T \tag{17.66}$$

where the vector $\mathbf{x}_i^k = [x_1^{k-1}, \ldots, x_{i-1}^{k-1}, x_i^k, x_{i+1}^{k-1}, \ldots, x_n^{k-1}]^T$ and $f(\mathbf{x}_i^k) = f(\mathbf{x}_1^{k-1}, \ldots, \mathbf{x}_{i-1}^{k-1}, \mathbf{x}_i^k, \mathbf{x}_{i+1}^{k-1}, \ldots, \mathbf{x}_n^{k-1})$. That is, only the ith variable is changed from $(k-1)$th to that of the kth iteration.

EXAMPLE 17.15

Continue with Example 17.14, except using the secant method for the line search. Recall that the objective function is

$$f(x_1, x_2) = \frac{(x_1 - 2)^2}{4} + (x_2 - 1)^2$$

We assume two points $\mathbf{x}^0 = [0, 0]^T$ and $\mathbf{x}^1 = [0.400, 0.800]^T$, and the first search direction $\mathbf{n}^1 = [0.841, 0.540]^T$ is given from Example 17.14. Therefore, we have $f(\mathbf{x}^0) = 2$ and $f(\mathbf{x}^1) = 0.680$.

Solution

For $k = 1$, we calculate $\mathbf{q}(\mathbf{x}^0, \mathbf{x}^1)$ following Eq. 17.66:

$$\mathbf{q}(\mathbf{x}^0, \mathbf{x}^1) = \begin{bmatrix} \frac{f(\mathbf{x}^0) - f(\mathbf{x}_1^1)}{x_1^0 - x_1^1} & \frac{f(\mathbf{x}^0) - f(\mathbf{x}_2^1)}{x_2^0 - x_2^1} \end{bmatrix}^T = \begin{bmatrix} \frac{f(0,0) - f(0.400, 0)}{0 - 0.400} & \frac{f(0,0) - f(0, 0.800)}{0 - 0.800} \end{bmatrix}^T = [-0.900 \quad -1.20]^T$$

Note that the analytical gradient vector at $\mathbf{x}^0$ is $\nabla f(\mathbf{x}^0) = [-1, -2]^T$. The approximation given above is not close because $\mathbf{x}^1$ is not close to $\mathbf{x}^0$. Now we calculate α^1 using Eq. 17.65b as

$$\alpha^1 = -\frac{f(\mathbf{x}^1)}{\mathbf{q}(\mathbf{x}^0, \mathbf{x}^1)^T \mathbf{n}^1} = -\frac{0.680}{[-0.900 \quad -1.20] \begin{bmatrix} 0.841 \\ 0.540 \end{bmatrix}} = 0.484$$

Hence

$\mathbf{x}^2 = \mathbf{x}^1 + \alpha^1 \mathbf{n}^1 = [0.400, 0.800]^T + 0.484\,[0.841, 0.540]^T = [0.407, 0.261]^T$. We then update the Hessian matrix starting from an identity matrix $\mathbf{H}^1 = \mathbf{I}$, and repeat the process until $||\nabla f(\mathbf{x}^k)|| < \varepsilon_2 = 0.001$.

The search method can be implemented in MATLAB to solve the example (see Script 17.11 for details). The optimal point is found at $\mathbf{x}^* = (2, 1)$, in which $f(\mathbf{x}^*) = 0$.

17.6 CONSTRAINED PROBLEMS*

The major difference between a constrained and an unconstrained problem is that for a constrained problem, an optimal solution must be sought in a feasible region; for an unconstrained problem, the feasible region contains the entire design domain. For a constrained problem, bringing an infeasible design into a feasible region is critical, in which gradients of active constraints are taken into consideration when determining the search direction for the next design. In this section, we first outline the nature of the constrained optimization problem and the concept of solution techniques. In Section 17.6.2, we then discuss a widely accepted strategy for dealing with the constraint functions, the so-called ε-active strategy. Thereafter, in Sections 17.6.3–17.6.5 we discuss the mainstream solution techniques for solving constrained optimization problems, including SLP, SQP, and the feasible direction method. These solution techniques are capable of solving general optimization problems with multiple constraints and many design variables. Before closing out this section, we introduce the penalty method, which solves a constrained problem by converting it to an unconstrained problem, and then we solve the unconstrained problem using methods discussed in Section 17.5. For illustration

purposes, we use simple examples of one or two design variables. Like Section 17.5, we offer sample MATLAB scripts for solving example problems.

17.6.1 BASIC CONCEPT

Recall the mathematical definition of the constrained optimization problem:

$$\text{Minimize:} \quad f(\mathbf{x}) \tag{17.67a}$$

$$\text{Subject to:} \quad g_i(\mathbf{x}) \leq 0, \quad i = 1,\ m \tag{17.67b}$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, p \tag{17.67c}$$

$$x_k^{\ell} \leq x_k \leq x_k^{u}, \quad k = 1, n \tag{17.67d}$$

Similar to solving unconstrained optimization problems, all numerical methods are based on the iterative process, in which the next design point is updated by a search direction $\mathbf{n}$ and a step size α along the direction. The next design point $\mathbf{x}^{k+1}$ is then obtained by evaluating the design at the current design point $\mathbf{x}^k$ (some methods include information from previous design iterations) as

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k = \mathbf{x}^k + \alpha^k \mathbf{n}^k \tag{17.68}$$

For an unconstrained problem, the search direction $\mathbf{n}$ considers only the gradient of the objective function. For constrained problems, however, optimal solutions must be sought in the feasible region. Therefore, active constraints in addition to objective functions must be considered while determining the search direction as well as the step size. As with the unconstrained problems, all algorithms need an initial design to initiate the iterative process. The difference is for a constrained problem, the starting design can be feasible or infeasible, as illustrated in Figure 17.15(a), in which a constrained optimization of two design variables x_1 and x_2 is assumed. The feasible region of the problem is identified on the surface of the objective function as well as projected onto the x_1–x_2 plane.

If an initial design is inside the feasible region, such as points A^0 or B^0, then we minimize the objective function by moving along its descent direction—say, the steepest descent direction—as if we are dealing with an unconstrained problem. We continue such iterations until either a minimum point is reached, such as the search path starting at point A^0, or a constraint becomes active (i.e., the boundary of the feasible region is reached, like the path of initial design at point B^0). Once the constraint boundary is encountered at point B^1, one strategy is to travel along a tangent line to the boundary, such as the direction B^1B^2 illustrated in Figure 17.15(b). This leads to an infeasible point from where the constraints are corrected in order to again reach the feasible point B^3. From there, the preceding steps are repeated until the optimum point is reached.

Another strategy is to deflect the tangential direction B^1B^2 toward the feasible region by a small angle θ when there are no equality constraints. Then, a line search is performed through the feasible region to reach the boundary point B^4, as shown in Figure 17.15(b). The procedure is then repeated from there.

When the starting point is infeasible, like points C^0 or D^0 in Figure 17.15(a), one strategy is to correct constraint violations to reach the constraint boundary. From there, the strategies described in the preceding paragraph can be followed to reach the optimum point. For example, for D^0, a similar path to that shown in path $B^1B^2B^3$ or B^1B^4 in Figure 17.15(b) is followed. The case for

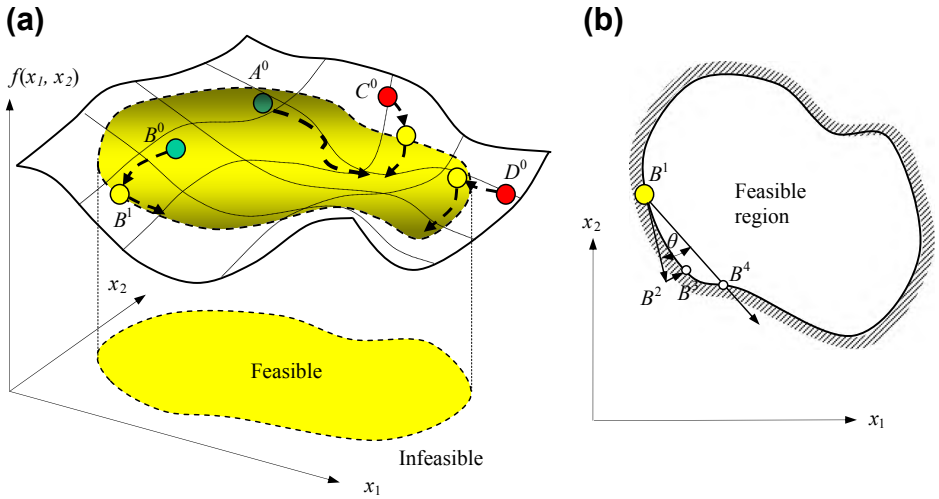


FIGURE 17.15

Concept illustration for solving a constrained optimization problem. (a) Paths illustrating different solution scenarios. (b) Top view of the feasible region, with a design point *B* residing on the boundary of the feasible region.

point C^0 is easier because the descent direction of objective function also corrects the constraint violations.

A good analogy for finding a minimum of a constrained problem is rolling a ball in a fenced downhill field. The boundary of the feasible region is the fence, and the surface of the downhill field is the objective function. When a ball is released at a location (i.e., the initial design), the ball rolls due to gravity. If the initial point is chosen such that the ball does not encounter the fence, the ball rolls to a local crest (minimum point). If an initial point chosen allows the ball to hit the fence, the ball rolls along the fence to reach a crest. If the initial point is outside the fenced area, the ball has to be thrown into the fenced area before it starts rolling.

Several algorithms based on the strategies described in the foregoing have been developed and evaluated. Some algorithms are better for a certain class of problems than others. In this section, we focus on general algorithms that have no restriction on the form of the objective or the constraint functions. Most of the algorithms that we will describe in this chapter can treat feasible and infeasible initial designs.

In general, numerical algorithms for solving constrained problems start with a linearization of the objective and constraint functions at the current design. The linearized subproblem is solved to determine the search direction **n**. Once the search direction is found, a line search is carried out to find an adequate step size α for the next design iteration. Following the general solution steps, we introduce three widely accepted methods: SLP, SQP, and the feasible direction method. Before we discuss the solution techniques, we discuss the ε-active strategy that determines the active constraints to incorporate for design optimization.

17.6.2 ε-ACTIVE STRATEGY

An ε-active constraint strategy (Arora, 2012), shown in Chapter 1, Figure 1.12, is often employed in solving constrained optimization problems. Inequality constraints in Eq. 17.67b and equality constraints of Eq. 17.67c are first normalized by their respective bounds:

$$b_i = \frac{g_i(\mathbf{x})}{g_i^u} - 1 \leq 0, \quad i = 1, m \tag{17.69a}$$

and

$$e_i = \frac{h_j(\mathbf{x})}{h_j^u} - 1, \quad j = 1, p \tag{17.69b}$$

Usually, when b_i (or e_i) is between two parameters CT (usually -0.03) and CTMIN (usually 0.005), g_i is active, as shown in Chapter 1, Figure 1.12. When b_i is less than CT, the constraint function is inactive or feasible. When b_i is larger than CTMIN, the constraint function is violated. Note that CTMIN$-$CT $= \varepsilon$.

17.6.3 THE SEQUENTIAL LINEAR PROGRAMMING ALGORITHM

The original optimization problem stated in Eq. 17.67 is first linearized by writing Taylor's expansions for the objective and constraint functions at the current design $\mathbf{x}^k$ as below.

Minimize the linearized objective function:

$$f\left(\mathbf{x}^{k+1}\right) = f\left(\mathbf{x}^k + \Delta\mathbf{x}^k\right) \approx f\left(\mathbf{x}^k\right) + \nabla f^{\mathrm{T}}\left(\mathbf{x}^k\right)\Delta\mathbf{x}^k \tag{17.70a}$$

subject to the linearized inequality constraints

$$g_i\left(\mathbf{x}^{k+1}\right) = g_i\left(\mathbf{x}^k + \Delta\mathbf{x}^k\right) \approx g_i\left(\mathbf{x}^k\right) + \nabla g_i^{\mathrm{T}}\left(\mathbf{x}^k\right)\Delta\mathbf{x}^k \leq 0; \quad i = 1, m \tag{17.70b}$$

and the linearized equality constraints

$$h_j\left(\mathbf{x}^{k+1}\right) = h_j\left(\mathbf{x}^k + \Delta\mathbf{x}^k\right) \approx h_j\left(\mathbf{x}^k\right) + \nabla h_j^{\mathrm{T}}\left(\mathbf{x}^k\right)\Delta\mathbf{x}^k = 0; \quad j = 1, p \tag{17.70c}$$

in which $\nabla f(\mathbf{x}^k)$, $\nabla g_i(\mathbf{x}^k)$, and $\nabla h_j(\mathbf{x}^k)$ are the gradients of the objective function, the ith inequality constraint and the jth equality constraint, respectively; and $\approx$ implies approximate equality.

To simplify the mathematical notations in our discussion, we rewrite the linearized equations in Eq. 17.70 as

$$\text{Minimize:} \quad \bar{f} = \mathbf{c}^{\mathrm{T}}\mathbf{d} \tag{17.71a}$$

$$\text{Subject to:} \quad \mathbf{A}^{\mathrm{T}}\mathbf{d} \leq \mathbf{b} \tag{17.71b}$$

$$\mathbf{N}^{\mathrm{T}}\mathbf{d} = \mathbf{e} \tag{17.71c}$$

$$-\Delta_\ell \leq \mathbf{d} \leq \Delta_u \tag{17.71d}$$

where

$$\mathbf{c}_{n\times 1} = \nabla f\left(\mathbf{x}^k\right) = \left[\partial f\left(\mathbf{x}^k\right)\Big/\partial x_1, \partial f\left(\mathbf{x}^k\right)\Big/\partial x_2, \ldots, \partial f\left(\mathbf{x}^k\right)\Big/\partial x_n\right]^{\mathrm{T}};$$

$$\mathbf{d}_{n\times 1} = \Delta\mathbf{x}^k = \left[\Delta x_1^k, \Delta x_2^k, \ldots, \Delta x_n^k\right]^{\mathrm{T}};$$

$$\mathbf{A}_{m\times n} = \left[\nabla g_1\left(\mathbf{x}^k\right), \nabla g_2\left(\mathbf{x}^k\right), \ldots, \nabla g_m\left(\mathbf{x}^k\right)\right]_{m\times n},$$

in which $$\nabla g_i\left(\mathbf{x}^k\right) = \left[\partial g_i\left(\mathbf{x}^k\right)\Big/\partial x_1, \partial g_i\left(\mathbf{x}^k\right)\Big/\partial x_2, \ldots, \partial g_i\left(\mathbf{x}^k\right)\Big/\partial x_n\right]_{n\times 1}^{\mathrm{T}};$$

$$\mathbf{N}_{p\times n} = \left[\nabla h_1\left(\mathbf{x}^k\right), \nabla h_2\left(\mathbf{x}^k\right), \ldots, \nabla h_p\left(\mathbf{x}^k\right)\right]_{p\times n},$$

in which $$\nabla h_i\left(\mathbf{x}^k\right) = \left[\partial h_i\left(\mathbf{x}^k\right)\Big/\partial x_1, \partial h_i\left(\mathbf{x}^k\right)\Big/\partial x_2, \ldots, \partial h_i\left(\mathbf{x}^k\right)\Big/\partial x_n\right]_{n\times 1}^{\mathrm{T}};$$

$$\mathbf{b}_{m\times 1} = -g_i\left(\mathbf{x}^k\right) = \left[-g_1\left(\mathbf{x}^k\right), -g_2\left(\mathbf{x}^k\right), \ldots, -g_m\left(\mathbf{x}^k\right)\right]_{m\times 1}^{\mathrm{T}};$$

and

$$\mathbf{e}_{m\times 1} = -h_i\left(\mathbf{x}^k\right) = \left[-h_1\left(\mathbf{x}^k\right), -h_2\left(\mathbf{x}^k\right), \ldots, -h_p\left(\mathbf{x}^k\right)\right]_{p\times 1}^{\mathrm{T}}.$$

Note that in Eq. 17.71a, $f(\mathbf{x}^k)$ is dropped. $\Delta_\ell = [\Delta_{1\ell}^k, \Delta_{2\ell}^k, \ldots, \Delta_{n\ell}^k]_{n\times 1}^{\mathrm{T}}$ and $\Delta_u = [\Delta_{1u}^k, \Delta_{2u}^k, \ldots, \Delta_{nu}^k]_{n\times 1}^{\mathrm{T}}$ are the move limits—that is, the maximum allowed decrease and increase in the design variables at the kth design iteration. Note that the move limits make the linearized subproblem bounded and give the design changes directly without performing the line search for a step size. Therefore, no line search is required in SLP. Choosing adequate move limits is critical to the SLP.

As discussed before, the SLP algorithm starts with an initial design $\mathbf{x}^0$. At the kth design iteration, we evaluate the objective and constraint functions as well as their gradients at the current design $\mathbf{x}^k$. We select move limits $\Delta_{i\ell}^k$ and Δ_{iu}^k to define an LP subproblem of Eq. 17.71. Solve the linearized subproblem for $\mathbf{d}^k$, and update the design for the next iteration as $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}^k$. The process repeats until convergent criteria are met. In general, the convergent criteria for an LP subproblem include

$$\overline{g}_i\left(\mathbf{x}^{k+1}\right) \le \varepsilon_1, \quad i = 1, m; \quad \left|\overline{h}_j\left(\mathbf{x}^{k+1}\right)\right| \le \varepsilon_1, \quad j = 1, p; \quad \text{and} \quad \left\|\mathbf{d}^k\right\| \le \varepsilon_2 \tag{17.72}$$

EXAMPLE 17.16

Solve Example 17.7 with one additional equality constraint using SLP. The optimization problem is restated as below

$$\text{Minimize:} \quad f(\mathbf{x}) = (x_1 - 3)^2 + (x_2 - 3)^2. \tag{17.73a}$$

$$\text{Subject to:} \quad g_1(\mathbf{x}) = 3x_1 + x_2 - 6 \le 0 \tag{17.73b}$$

$$h_1(\mathbf{x}) = x_1 - x_2 = 0 \tag{17.73c}$$

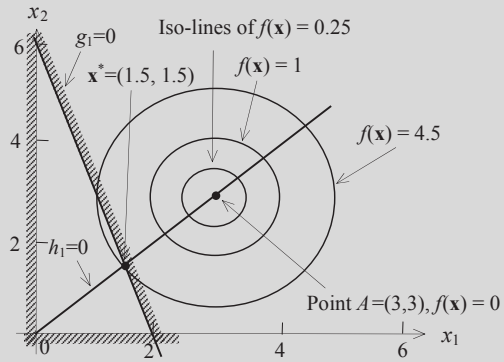
$$0 \le x_1, 0 \le x_2 \tag{17.73d}$$

Continued

EXAMPLE 17.16—cont'd

Solution

We sketch the feasible region bounded by inequality constraint $g_1(\mathbf{x}) \leq 0$, side constraints, and equality constraint $h_1(\mathbf{x}) = 0$, as shown below. As is obvious in the sketch, the optimal solution is found at $\mathbf{x}^* = (1.5, 1.5)$, the intersection of $g_1(\mathbf{x}) = 0$, and $h_1(\mathbf{x}) = 0$, in which $f(\mathbf{x}) = 4.5$.



Now we use this example to illustrate the solution steps using SLP. We assume an initial design at $\mathbf{x}^0 = (2, 2)$. At the initial design, we have $f(2, 2) = (x_1 - 3)^2 + (x_2 - 3)^2 = 2$, $g_1(2, 2) = 3x_1 + x_2 - 6 = 2 > 0$, and $h_1(2, 2) = 0$. The inequality constraint g_1 is greater than 0; therefore, this constraint is violated. The initial design is not in the feasible region, as also illustrated in the figure above. The optimization problem defined in Eqs 17.73a—17.73d is linearized as follows:

$$\text{Minimize:} \quad \overline{f} = \mathbf{c}^{\mathrm{T}}\mathbf{d} = [2(x_1 - 3) \quad 2(x_2 - 3)]\begin{bmatrix}\Delta x_1 \\ \Delta x_2\end{bmatrix} \tag{17.73e}$$

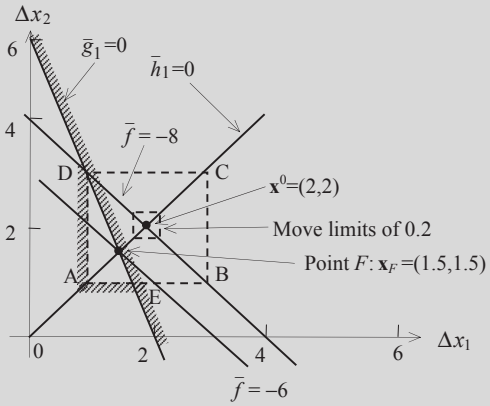
$$\text{Subject to:} \quad \mathbf{A}^{\mathrm{T}}\mathbf{d} \leq \mathbf{b};\ \text{i.e.,}\ [3 \quad 1]\begin{bmatrix}\Delta x_1 \\ \Delta x_2\end{bmatrix} \leq 6, \quad \text{or} \quad \overline{g}_1 = 3\Delta x_1 + \Delta x_2 - 6 \leq 0 \tag{17.73f}$$

$$\mathbf{N}^{\mathrm{T}}\mathbf{d} = \mathbf{e};\ \text{i.e.,}\ [1 \quad -1]\begin{bmatrix}\Delta x_1 \\ \Delta x_2\end{bmatrix} = 0, \quad \text{or} \quad \overline{h}_1 = \Delta x_1 - \Delta x_2 = 0 \tag{17.73g}$$

$$-0.2 \leq \Delta x_1 \leq 0.2, -0.2 \leq \Delta x_2 \leq 0.2 \tag{17.73h}$$

We have chosen the move limits to be 0.2, which is 10% of the current design variable values, as shown in Eq. 17.73h. At the initial design, $\mathbf{x}^0 = (2, 2)$, we are minimizing $\overline{f} = [-2 \quad -2]\begin{bmatrix}\Delta x_1 \\ \Delta x_2\end{bmatrix} = -2\Delta x_1 - 2\Delta x_2$ subject to constraints (Eqs 17.73f—17.73h). The subproblem has two variables; it can be solved by referring to the sketch below. Because we chose 0.2 as the move limits, the solution to the LP subproblem must lie in the region of the small dotted square box shown below. It can be seen that there is no feasible solution to this linearized subproblem because the small box does not intersect the line $\overline{g}_1 = 0$. We must enlarge this region by increasing the move limits. Thus, we note that if the move limits are too restrictive, the linearized subproblem may not have a solution.

EXAMPLE 17.16—cont'd



If we choose the move limits to be 1—that is, 50% of the design variable values—then the design must lie within a larger box ABCD of 2×2 as shown above. Hence the feasible region of the LP problem is now the triangle AED intersecting $\bar{h}_1 = 0$ (that is, line segment AF). Therefore, the optimal solution of the LP problem is found at point F: $\mathbf{x}_F = (1.5, 1.5)$, where $\bar{f} = -6$. That is, $\mathbf{d} = [\Delta x_1^0, \Delta x_2^0]^T = [-0.5, -0.5]^T$, and $\mathbf{x}^1 = \mathbf{x}^0 + \mathbf{d} = [2, 2]^T + [-0.5, -0.5]^T = [1.5, 1.5]^T = \mathbf{x}_F$.

In the next design $\mathbf{x}^1$, we evaluate the objective and constraint functions of the original optimization problem as well as their gradients. We have $f(1.5, 1.5) = (x_1 - 3)^2 + (x_2 - 3)^2 = 4.5$, $g_1(1.5, 1.5) = 3x_1 + x_2 - 6 = 0$, and $h_1(1.5, 1.5) = 0$. The design is feasible. Again, at the design iteration $\mathbf{x}^1 = (1.5, 1.5)$, we create the LP problem as

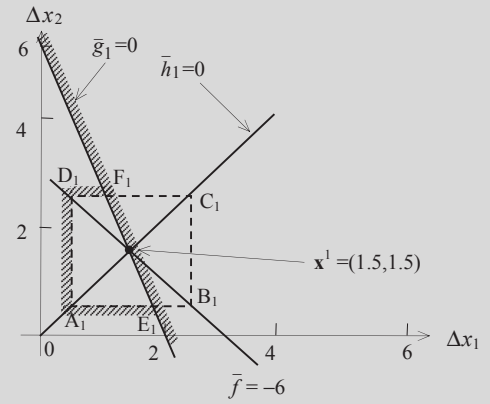
$$\text{Minimizing:} \quad \bar{f} = [-3 \quad -3] \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = -3\Delta x_1 - 3\Delta x_2$$

$$\text{Subject to:} \quad \bar{g}_1 = 3\Delta x_1 + \Delta x_2 - 6 \le 0$$

$$\bar{h}_1 = \Delta x_1 - \Delta x_2 = 0$$

$$-1 \le \Delta x_1 \le 1, -1 \le \Delta x_2 \le 1$$

As illustrated in the figure next page, the feasible region of the LP subproblem is now the polygon $A_1E_1F_1D_1$ intersecting $\bar{h}_1 = 0$. Therefore, the optimal solution of the LP problem is found again at $\mathbf{x}^1 = (1.5, 1.5)$, the same point. That is, in this design iteration, $\mathbf{d} = [\Delta x_1^1, \Delta x_2^1]^T = [0, 0]^T$.



Continued

EXAMPLE 17.16—cont'd

At this point, the convergent criterion stated in Eq. 17.72, for example, $||\mathbf{d}^1|| = 0 \leq \varepsilon_2$, is satisfied; hence, an optimal solution is found at $\mathbf{x}^1 = (1.5, 1.5)$.

In fact, for this particular problem, it takes only one iteration to find the optimal solution. In general, this may not be the case. An iterative process often takes numerous iterations to achieve a convergent solution.

Although the SLP algorithm is a simple and straightforward approach to solving constrained optimization problems, it should not be used as a black-box approach for engineering design problems. The selection of move limits is in essence trial and error and can be best achieved in an interactive mode. Also, the method may not converge to the precise minimum because no descent function is defined, and the line search is not performed along the search direction to compute a step size. Nevertheless, this method may be used to obtain improved designs in practice. It is a good method to include in our toolbox for solving constrained optimization problems.

17.6.4 THE SEQUENTIAL QUADRATIC PROGRAMMING ALGORITHM

The SQP algorithm incorporates second-order information about the problem functions in determining a search direction $\mathbf{n}$ and step size α. A search direction in the design space is calculated by utilizing the values and the gradients of the objective and constraint functions. A quadratic programming subproblem is defined as

$$\text{Minimize:} \quad \overline{f} = \mathbf{c}^{\mathrm{T}}\mathbf{d} + \frac{1}{2}\mathbf{d}^{\mathrm{T}}\mathbf{d} \tag{17.74a}$$

$$\text{Subject to:} \quad \mathbf{A}^{\mathrm{T}}\mathbf{d} \leq \mathbf{b} \tag{17.74b}$$

$$\mathbf{N}^{\mathrm{T}}\mathbf{d} = \mathbf{e} \tag{17.74c}$$

in which a quadratic term is added to the objective function $\overline{f}$ and the constraint functions (17.74b) and (17.74c) are identical to those of the LP subproblem, except that there is no need to define the move limits. The solution of the QP problem $\mathbf{d}$ defines the search direction $\mathbf{n}$ (where $\mathbf{n} = \mathbf{d}/||\mathbf{d}||$). Once the search direction is determined, a line search is carried out to find an adequate step size α. The process repeats until the convergent criteria defined in Eq. 17.72 are met.

EXAMPLE 17.17

Solve the same problem of Example 17.16 using SQP.

Solution

We assume the same initial design at $\mathbf{x}^0 = (2, 2)$. At the initial design, we have $f(2, 2) = (x_1 - 3)^2 + (x_2 - 3)^2 = 2$, $g_1(2, 2) = 3x_1 + x_2 - 6 = 2 > 0$, and $h_1(2, 2) = 0$. The initial design is infeasible. The QP subproblem can be written as

$$\text{Minimize:} \quad \overline{f} = \mathbf{c}^{\mathrm{T}}\mathbf{d} + \frac{1}{2}\mathbf{d}^{\mathrm{T}}\mathbf{d} = [\,2(x_1 - 3) \quad 2(x_2 - 3)\,]\begin{bmatrix}\Delta x_1 \\ \Delta x_2\end{bmatrix} + \frac{1}{2}[\,\Delta x_1 \quad \Delta x_2\,]\begin{bmatrix}\Delta x_1 \\ \Delta x_2\end{bmatrix} \tag{17.75a}$$

$$\text{Subject to:} \quad \overline{g}_1 = 3\Delta x_1 + \Delta x_2 - 6 \leq 0 \tag{17.75b}$$

$$\overline{h}_1 = \Delta x_1 - \Delta x_2 = 0 \tag{17.75c}$$

EXAMPLE 17.17—cont'd

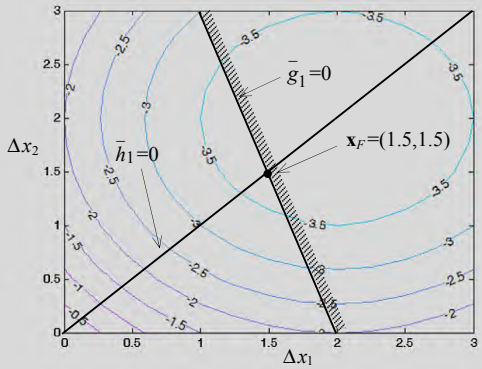
At the initial design, $\mathbf{x}^0 = (2, 2)$, we are minimizing

$$\bar{f} = [-2 \quad -2]\begin{bmatrix}\Delta x_1 \\ \Delta x_2\end{bmatrix} + \frac{1}{2}[\Delta x_1 \quad \Delta x_2]\begin{bmatrix}\Delta x_1 \\ \Delta x_2\end{bmatrix} = -2\Delta x_1 - 2\Delta x_2 + \frac{1}{2}(\Delta x_1^2 + \Delta x_2^2)$$

subject to constraint Eqs 17.75b and 17.75c. The QP subproblem can be solved by either using the KKT condition or graphical method. We use the graphical method for this example.

Referring to the sketch below, the optimal solution to the QP subproblem is found at point F: $\mathbf{x}_F = (1.5, 1.5)$, where $\bar{f} = -3.75$. That is, $\mathbf{d} = [\Delta x_1^0, \Delta x_2^0]^{\mathrm{T}} = [-0.5, \ -0.5]^{\mathrm{T}}$. Note that the quadratic function $\bar{f}$ is sketched using the MATLAB script (Script 17.12).

For the next iteration, we have $\mathbf{n} = \mathbf{d}/\|\mathbf{d}\| = [-0.707, -0.707]^{\mathrm{T}}$ and assume a step size $\alpha = 1$. Hence, for the next design, $\mathbf{x}^1 = \mathbf{x}^0 + \alpha\mathbf{n}^0 = [2, 2]^{\mathrm{T}} + 1[-0.707, -0.707]^{\mathrm{T}} = [1.293, 1.293]^{\mathrm{T}}$.



In the next design $\mathbf{x}^1$, we evaluate the objective and constraint functions of the original optimization problem as well as their gradients. We have $f(1.293, 1.293) = (x_1 - 3)^2 + (x_2 - 3)^2 = 5.828$, $g_1(1.293, 1.293) = 3x_1 + x_2 - 6 = -0.828$, and $h_1(1.293, 1.293) = 0$. The design is feasible. Again, at the design iteration $\mathbf{x}^1 = (1.293, 1.293)$, we create the QP problem as

$$\text{Minimizing:} \quad \bar{f} = [-3.414 \ -3.414]\begin{bmatrix}\Delta x_1 \\ \Delta x_2\end{bmatrix} + \frac{1}{2}[\Delta x_1 \ \Delta x_2]\begin{bmatrix}\Delta x_1 \\ \Delta x_2\end{bmatrix} = -3.414\Delta x_1 - 3.414\Delta x_2 + \frac{1}{2}(\Delta x_1^2 + \Delta x_2^2)$$

$$\text{Subject to:} \quad \bar{g}_1 = 3\Delta x_1 + \Delta x_2 - 6 \le 0$$

$$\bar{h}_1 = \Delta x_1 - \Delta x_2 = 0$$

The optimal design of the QP problem is found again at $\mathbf{x}^1 = (1.5, 1.5)$, the same point since the constraint functions are unchanged. That is, $\mathbf{d} = [\Delta x_1^1, \Delta x_2^1]^{\mathrm{T}} = [0.207, 0.207]^{\mathrm{T}}$. Therefore, the convergent criterion stated in Eq. 17.72 are satisfied, and an optimal solution is found at $\mathbf{x}^* = (1.5, 1.5)$.

17.6.5 FEASIBLE DIRECTION METHOD

The basic idea of the feasible direction method is to determine a search direction that moves from the current design point to an improved feasible point in the design space. Thus, given a design $\mathbf{x}^k$, an

improving feasible search direction $\mathbf{n}^k$ is determined such that for a sufficiently small step size $\alpha > 0$, the new design, $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{n}^k$ is feasible, and the new objective function is smaller than the current one; that is, $f(\mathbf{x}^{k+1}) < f(\mathbf{x}^k)$. Note that $\mathbf{n}$ is a normalized vector, defined as $\mathbf{n} = \mathbf{d}/||\mathbf{d}||$, where $\mathbf{d}$ is the nonnormalized search direction solved from a subproblem to be discussed.

Because along the search direction $\mathbf{d}^k$ the objective function must decrease without violating the applied constraints, taking into account only inequality constraints, it must result that

$$\nabla f\left(\mathbf{x}^k\right)^{\mathrm{T}} \cdot \mathbf{d}^k < 0 \tag{17.76}$$

and

$$\nabla g_i\left(\mathbf{x}^k\right)^{\mathrm{T}} \cdot \mathbf{d}^k < 0, \quad \text{for } i \in \mathrm{I}_k \tag{17.77}$$

I_k is the potential constraint set at the current point, defined as

$$\mathrm{I}_k \equiv \left\{ i \middle| g_i(\mathbf{x})^k + \varepsilon \geq 0 \;\; i = 1, m \right\} \tag{17.78}$$

Note that ε is a small positive number, selected to determine ε-active constraints as discussed in Section 17.6.1. Note that $g_i(\mathbf{x})$ is normalized as in Eq. 17.69a. The inequality constraints enclosed in the set of Eq. 17.78 are either violated or ε-active, meaning they have to be considered in determining a search direction that brings the design into the feasible region. Equations 17.76 and 17.77 are referred to as usability and feasibility requirements, respectively. A geometrical interpretation of the requirements is shown in Figure 17.16 for a two-variable optimization problem, in which the search direction $\mathbf{n}$ points to the usable-feasible region.

This method has been developed and applied mostly to optimization problems with inequality constraints. This is because, in implementation, the search direction $\mathbf{n}$ is determined by defining a linearized subproblem (to be discussed next) at the current feasible point, and the step size α is determined to reduce the objective function as well as maintain feasibility of design. Because linear approximations are used, it is difficult to maintain feasibility with respect to the equality constraints. Although some procedures have been developed to treat equality constraints in these methods, we will describe the method for problems with only inequality constraints.

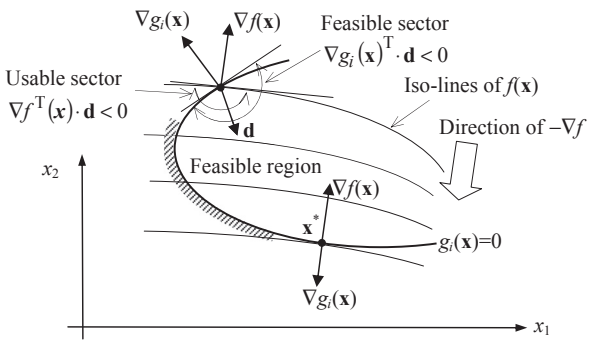


FIGURE 17.16

Geometric description of the feasible direction method.

The desired search direction $\mathbf{d}$ will meet the requirements of usability and feasibility, and it gives the highest reduction of the objective function along it. Mathematically, it is obtained by solving the following linear subproblem in $\mathbf{d}$:

$$\text{Minimize:} \quad \beta \tag{17.79a}$$

$$\text{Subject to:} \quad \nabla f^{\mathrm{T}}(\mathbf{x})\mathbf{d} - \beta \leq 0 \tag{17.79b}$$

$$\nabla g_i^{\mathrm{T}}(\mathbf{x})\mathbf{d} - \beta \leq 0, \quad \text{for } i \in \mathrm{I}_k \tag{17.79c}$$

$$\mathrm{d}_j^{\ell} \leq \mathrm{d}_j \leq \mathrm{d}_j^{u}, \quad \text{for } j = 1, n \tag{17.79d}$$

Note that this is a linear programming problem. If $\beta < 0$, then $\mathbf{d}$ is an improving feasible direction. If $\beta = 0$, then the current design satisfies the KKT necessary conditions and the optimization process is terminated. To compute the improved design in this direction, a step size α is needed.

EXAMPLE 17.18

Solve the optimization problem of Example 17.7 using the feasible direction method. The problem is restated below:

$$\text{Minimize:} \quad f(\mathbf{x}) = (x_1 - 3)^2 + (x_2 - 3)^2 \tag{17.80a}$$

$$\text{Subject to:} \quad g_1(\mathbf{x}) = 3x_1 + x_2 - 6 \leq 0 \tag{17.80b}$$

$$0 \leq x_1, \quad \text{and} \quad 0 \leq x_2 \tag{17.80c}$$

Solution

Referring to the sketch of Example 17.7, the optimal solution is found as $\mathbf{x}^* = (1.2, 2.4)$, in which $f(\mathbf{x}) = 3.6$. In this example, we present two cases of two respective initial designs, one in the feasible region and the other one in the infeasible region.

Case A: feasible initial design at $\mathbf{x^0} = (1, 1)$. From Eq. 17.79, a subproblem can be written at the initial design as

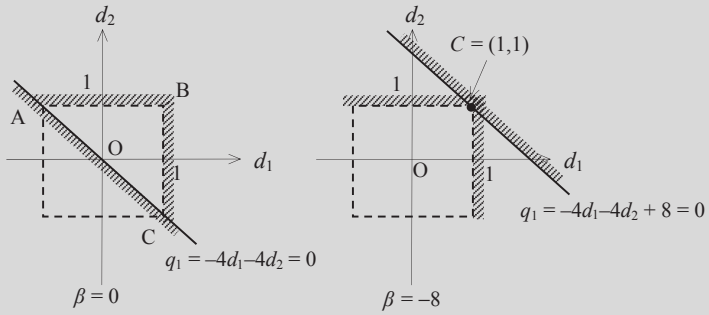
$$\text{Minimize:} \quad \beta \tag{17.80d}$$

$$\text{Subject to:} \quad q_1 = -4d_1 - 4d_2 - \beta \leq 0 \tag{17.80e}$$

$$-1 \leq d_1 \leq 1, -1 \leq d_2 \leq 1 \tag{17.80f}$$

Note that we do not need to include the linearized constraint equation $g_1 \leq 0$ because the design is feasible. We assume lower and upper bounds of the subproblem as -1 and 1, respectively, as stated in Eq. 17.80f.

We sketch the feasible region defined by Eqs 17.80e and 17.80f with $\beta = 0$ and -8, respectively, as shown below. For $\beta = 0$, the feasible region is the triangle ABC, and for $\beta = -8$, the feasible region reduces to a single point $C = (1, 1)$. As is obvious in the sketches below, the optimal solution of the subproblem is found at $C = (1, 1)$, in which $\beta = -8$.



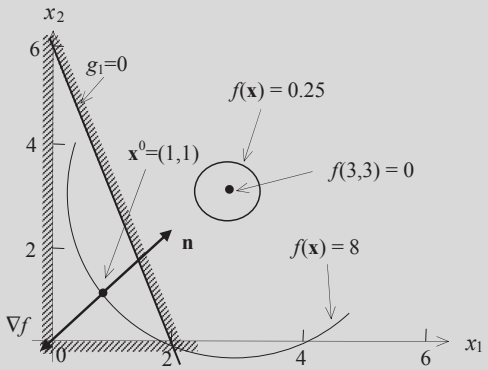
Continued

EXAMPLE 17.18—cont'd

Certainly, if the bounds of d_1 and d_2 are changed, the solution changes as well. However, the search direction defined by $\mathbf{n} = \mathbf{d}/||\mathbf{d}|| = [1, 1]^T/||[1, 1]^T|| = [0.707, 0.707]^T$ remains the same. From Eq. 17.79b, we have

$$\nabla f^T(1, 1)\mathbf{d} = [-4, -4][1, 1]^T = -8(= \beta) < 0$$

Note that $\nabla f^T \mathbf{d}$ is a dot product of ∇f^T and $\mathbf{d}$. Geometrically, $(\nabla f^T \mathbf{d})/||\nabla f^T \mathbf{d}|| = -1$ is the angle between the vectors ∇f^T and $\mathbf{d}$—in this case 180°, as shown below. This is because the initial design is feasible, and the search direction $\mathbf{n}$ (or $\mathbf{d}$) is the negative of the gradient of the objective function ∇f.



Case B: infeasible initial design at $\mathbf{x}^0 = (2, 2)$. From Eq. 17.79, a subproblem can be written at the initial design as

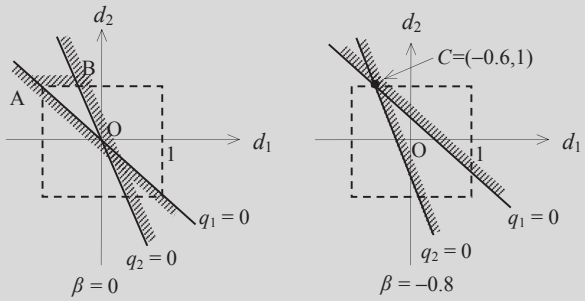
$$\text{Minimize:} \quad \beta \tag{17.80g}$$

$$\text{Subject to:} \quad q_1 = -2d_1 - 2d_2 - \beta \le 0 \tag{17.80h}$$

$$q_2 = 3d_1 + d_2 - \beta \le 0 \tag{17.80i}$$

$$-1 \le d_1 \le 1, \quad -1 \le d_2 \le 1 \tag{17.80j}$$

Similar to Case A, we sketch the feasible region defined by Eqs 17.80h–17.80j with $\beta = 0$ and -0.8, respectively, as shown below. For $\beta = 0$, the feasible region is the triangle ABO, and for $\beta = -0.8$, the feasible region reduces to a single point at $C = (-0.6, 1)$. As is obvious in the sketches, the optimal solution of the subproblem is found at $C = (-0.6, 1)$, in which $\beta = -0.8$.

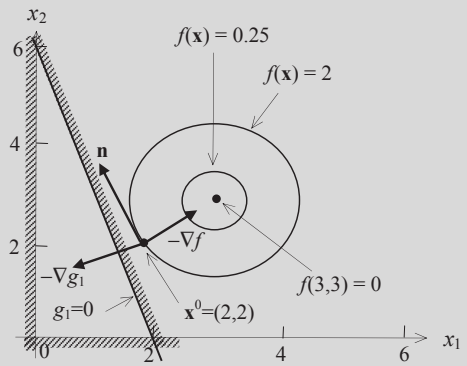


EXAMPLE 17.18—cont'd

The search direction is found as $\mathbf{n} = \mathbf{d}/\|\mathbf{d}\| = [-0.6, 1]^{\mathrm{T}}/\|[-0.6, 1]\|^{\mathrm{T}} = [-0.441, 0.735]^{\mathrm{T}}$. From Eqs 17.79b and 17.79c, we have

$$\nabla f^{\mathrm{T}}(2,2)\mathbf{d} = [-2, -2][-0.6, 1]^{\mathrm{T}} = -0.8(=\beta) < 0, \quad \text{and} \quad \nabla g_i^{\mathrm{T}}(2,2)\mathbf{d} = [3, 1][-0.6, 1]^{\mathrm{T}} = -0.8 < 0$$

Geometrically, they are the respective angles between the vectors ∇f^{T} and $\mathbf{d}$ and ∇g_i^{T} and $\mathbf{d}$, as shown below. Because the design is infeasible, the gradient of the active constraint is taken into consideration in calculating the search direction. In fact, because the same parameter β is employed in the constraint equations of the subproblem, the search direction $\mathbf{n}$ points in a direction that splits the angle between $-\nabla f$ and $-\nabla g_1$.



In the constraint equations of the subproblem stated in Eqs 17.79b and 17.79c, the same parameter β is employed. As demonstrated in Case B of Example 17.18, the same β leads to a search direction $\mathbf{n}$ pointing in a direction that splits the angle between $-\nabla f$ and $-\nabla g_1$, in which g_1 is an active constraint function.

To determine a better feasible direction $\mathbf{d}$, the constraints of Eq. 17.79c can be modified as

$$\nabla g_i^{\mathrm{T}}(\mathbf{x})\mathbf{d} - \theta_i\beta \le 0, \quad \text{for } i \in \mathrm{I}_K \tag{17.81}$$

where θ_i is called the push-off factor. The greater the value of θ_i, the more the direction vector $\mathbf{d}$ is pushed into the feasible region. The reason for introducing θ_i is to prevent the iterations from repeatedly hitting the constraint boundary and slowing down the convergence.

EXAMPLE 17.19

Find the search directions $\mathbf{n}$ for Case B of Example 17.18, assuming $\theta_1 = 0$, 0.5, and 1.5.

Solution

We show the solutions in the following four cases.

Case A: $\theta_1 = 0$. From Eqs 17.79 and 17.81, a subproblem can be written at the initial design as

$$\text{Minimize:} \quad \beta \tag{17.82a}$$

$$\text{Subject to:} \quad q_1 = -2d_1 - 2d_2 - \beta \le 0 \tag{17.82b}$$

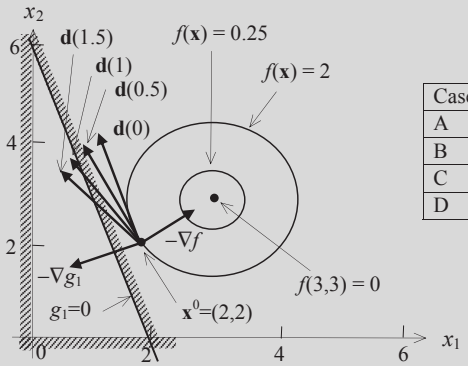
$$q_2 = 3d_1 + d_2 \le 0 \tag{17.82c}$$

$$-1 \le d_1 \le 1, \quad -1 \le d_2 \le 1 \tag{17.82d}$$

Continued

EXAMPLE 17.19—cont'd

Following the similar approach as in Example 17.18, the solution to the subproblem defined in Eqs 17.82a–17.82d is found at $\mathbf{d} = (-1/3, 1)$ with $\beta = -4/3$. In fact, the search direction points in a direction that is parallel to the active constraint g_1 at the current design $\mathbf{x}^0$; i.e., the search direction is perpendicular to the gradient of the constraint function ∇g_1, as shown below in the vector $\mathbf{d}(0)$.



Case	θ_1	$\mathbf{d}$	β
A	0	$\mathbf{d}(0) = (-1/3,1)$	−4/3
B	0.5	$\mathbf{d}(0.5) = (-1/2,1)$	−1
C	1	$\mathbf{d}(1) = (-0.6,1)$	−0.8
D	1.5	$\mathbf{d}(1.5) = (-2/3,1)$	−2/3

Case B: $\theta_1 = 0.5$. The constraint equation q_2 of the subproblem becomes

$$q_2 = 3d_1 + d_2 - 0.5\beta \leq 0 \tag{17.82e}$$

The solution to the subproblem is found at $\mathbf{d} = (-1/2, 1)$ with $\beta = -1$. The search direction $\mathbf{n}$ leans to $-\nabla g_1$, as shown in the figure above in the vector $\mathbf{d}(0.5)$. That is, the design is pushed more into the feasible region compared to the case where $\theta_1 = 0$.

Case C: $\theta_1 = 1$. The constraint equation q_2 of the subproblem becomes

$$q_2 = 3d_1 + d_2 - \beta \leq 0 \tag{17.82f}$$

The solution to the subproblem is found at $\mathbf{d} = (-0.6, 1)$ with $\beta = -0.8$. The search direction $\mathbf{d}$ leans more to $-\nabla g_1$, as shown in the figure above in the vector $\mathbf{d}(1)$.

Case D: $\theta_1 = 1.5$. The constraint equation q_2 of the subproblem becomes

$$q_2 = 3d_1 + d_2 - 1.5\beta \leq 0 \tag{17.82g}$$

The solution to the subproblem is found at $\mathbf{d} = (-2/3, 1)$ with $\beta = -2/3$. The search direction $\mathbf{d}$ leans more to $-\nabla g_1$, as shown in the figure above in the vector $\mathbf{d}(1.5)$.

17.6.6 PENALTY METHOD

A penalty method replaces a constrained optimization problem by a series of unconstrained problems whose solutions ideally converge to the solution of the original constrained problem. The unconstrained problems are formed by adding a term, called a penalty function, to the objective function that consists of a penalty parameter multiplied by a measure of violation of the constraints. The measure of violation is nonzero when the constraints are violated and is zero in the region where constraints are not violated.

Recall that a constrained optimization problem considered is defined as

$$\underset{\mathbf{x}\in S}{\text{Minimize}} \quad f(\mathbf{x}) \tag{17.83}$$

where S is the set of feasible designs defined by equality and inequality constraints. Using the penalty method, Eq. 17.83 is first converted to an unconstrained problem as

$$\text{Minimize} \quad \Phi(\mathbf{x}, r_p) = f(\mathbf{x}) + r_p p(\mathbf{x}) \tag{17.84}$$

where $f(\mathbf{x})$ is the original objective function, $p(\mathbf{x})$ is an imposed penalty function, and r_p is a multiplier that determines the magnitude of the penalty. The function $\Phi(\mathbf{x}, r_p)$ is called pseudo-objective function.

There are numerous ways to create a penalty function. One of the easiest is called exterior penalty (Vanderplaats, 2007), in which a penalty function is defined as

$$p(\mathbf{x}) = \sum_{i=1}^{m} \left\{\max\left[0, g_i(\mathbf{x})\right]\right\}^2 + \sum_{j=1}^{p} \left[h_j(\mathbf{x})\right]^2 \tag{17.85}$$

From Eq. 17.85, we see that no penalty is imposed if all constraints are satisfied. However, whenever one or more constraints are violated, the square of these constraints is included in the penalty function.

If we choose a small value for the multiplier r_p, the pseudo-objective function $\Phi(\mathbf{x}, r_p)$ may be solved easily, but may converge to a solution with large constraint violations. On the other hand, a large value of r_p ensures near satisfaction of all constraints but may create a poorly conditioned optimization problem that is unstable and difficult to solve numerically. Therefore, a better strategy is to start with a small r_p and minimize $\Phi(\mathbf{x}, r_p)$. Then, we increase r_p by a factor of γ (say $\gamma = 10$), and proceed with minimizing $\Phi(\mathbf{x}, r_p)$ again. Each time, we take the solution from the previous optimization problem as the initial design to speed up the optimization process. We repeat the steps until a satisfactory result is obtained. In general, solutions to the successive unconstrained problems will eventually converge to the solution of the original constrained problem.

EXAMPLE 17.20

Solve the following optimization problem using the penalty method.

$$\text{Minimize:} \quad f(x) = x \tag{17.86a}$$

$$\text{Subject to:} \quad g_1(x) = 1 - x \leq 0 \tag{17.86b}$$

$$g_2(x) = \frac{1}{2}x - 1 \leq 0 \tag{17.86c}$$

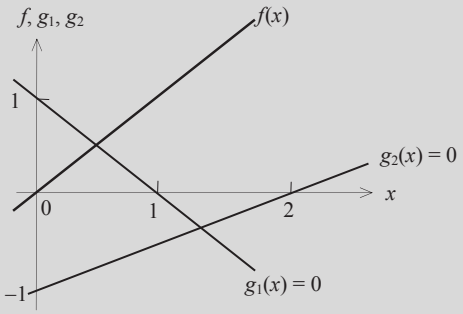
$$0 \leq x \leq 3 \tag{17.86d}$$

Continued

EXAMPLE 17.20—cont'd

Solution

We show the objective and constraint functions in the sketch below. It is obvious that the feasible region is [1, 2], and the optimal solution is at $x = 1, f(1) = 1$.



Now we solve the problem using the penalty method. We convert the constrained problem to an unconstrained problem using Eq. 17.84 as

$$\text{Minimize } \Phi(x, r_p) = x + r_p\left\{[\max(0, 1 - x)]^2 + [\max(0, 0.5x - 1)]^2\right\} \tag{17.86e}$$

We start with $r_p = 1$ and use the golden search to find the solution. The MATLAB script for finding the solution to Eq. 17.86e can be found in Script 17.13.

For $r_p = 1$, the golden search found a solution of $x = 0.5$, $\Phi(0.5, 1) = 0.75$, with constraint functions $g_1(0.5) = 0.5$ (violated) and $g_2(0.5) = -0.75$ (satisfied).

We increase $r_p = 10$. The golden search found a solution of $x = 0.950$, $\Phi(0.950, 10) = 0.975$, with constraint functions $g_1(0.950) = 0.05$ (violated) and $g_2(0.950) = -0.525$ (satisfied).

We increase $r_p = 100$. The golden search found a solution of $x = 0.995$, Φ $(0.995, 100) = 0.9975$, with constraint functions $g_1(0.995) = 0.005$ (violated) and $g_2(0.995) = -0.5025$ (satisfied).

When we increase $r_p = 10{,}000$, we have $x = 0.9999$, $\Phi(0.9999, 100) = 1.000$, with constraint functions $g_1(0.9999) = 0.00001$ (violated) and $g_2(0.9999) = -0.500$ (satisfied). At this point, the objective function is $f(0.9999) = 0.9999$. The convergent trend is clear from results of increasing the r_p value.

17.7 NON-GRADIENT APPROACH*

Unlike the gradient-based approach, the non-gradient approach uses only the function values in the search process, without the need for gradient information. The algorithms developed in this approach are very general and can be applied to all kinds of problems—discrete, continuous, or non-differentiable functions. In addition, the methods determine global optimal solutions as opposed to the local optimal determined by a gradient-based approach. Although the non-gradient approach does not require the use of gradients of objective or constraint functions, the solution techniques require a large amount of function evaluations. For large-scale problems that require significant computing time for function evaluations, the non-gradient approach is too expensive to use. Furthermore, there is no guarantee that a global optimum can be obtained. The computation issue may be overcome to some extent by the use of parallel computing or supercomputers. The issue of global optimal solution may be overcome to some extent by allowing the algorithm to run longer.

In this section, we discuss two popular and representative algorithms of the non-gradient approach: the genetic algorithm (GA) and simulated annealing (SA). We introduce concepts and solution process for the algorithms to provide readers with a basic understanding of these methods.

17.7.1 GENETIC ALGORITHMS

Recall that the optimization problem considered is defined as

$$\underset{\mathbf{x}\in S}{\text{Minimize}} \quad f(\mathbf{x}) \tag{17.87}$$

where S is the set of feasible designs defined by equality and inequality constraints. For unconstrained problems, S is the entire design space. Note that to use a genetic algorithm, the constrained problem is often converted to an unconstrained problem using, for example, the penalty method discussed in Section 17.6.6.

17.7.1.1 Basic Concepts

A genetic algorithm starts with a set of designs or candidate solutions to a given optimization problem and moves toward the optimal solution by applying the mechanism mimicking evolution principle in nature—that is, survival of the fittest. The set of candidate solutions is called a population. A candidate solution in a population is called individual, creature, or phenotype. Each candidate solution has a set of properties represented in its chromosomes (or genotype) that can be mutated and altered. Traditionally, candidate solutions are represented in binary strings of 0 and 1, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals. The solution process is iterative, with the population in each iteration called a generation. The size of the population in each generation is unchanged. In each generation, the fitness of every individuals in the population is evaluated; the fitness is usually the value of the objective (or pseudo-objective converted from a constrained problem) function in the optimization problem being solved. The better fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and possibly randomly mutated) to form a new generation. Because better fit members of the set are used to create new designs, the successive generations have a higher probability of having designs with better fitness values. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been reached or a satisfactory fitness level has been achieved for the population.

17.7.1.2 Design Representation

In a genetic algorithm, an individual (or a design point) consists of a chromosome and a fitness function.

The chromosome represents a design point, which contains values for all the design variables of the design problem. The gene represents the value of a particular design variable. The simplest algorithm represents each chromosome as a bit string. Typically, an encoding scheme is prescribed, in which numeric parameters can be represented by integers, although it is possible to use floating point representations as well. The basic algorithm performs crossover and mutation at the bit level. For example, we assume an optimization problem of three design variables: $\mathbf{x} = [x_1, x_2, x_3]^{\mathrm{T}}$. We use a string length of 4 (or 4 bits) for each design variable, in which $2^4 = 16$ discrete values can be

represented. If, at the current design, the design variables are $x_1 = 4$, $x_2 = 7$, and $x_3 = 1$, the chromosome length is $C = 12$:

$$\underbrace{0\,1\,0\,0}_{x_1=4}\ \underbrace{0\,1\,1\,1}_{x_2=7}\ \underbrace{0\,0\,0\,1}_{x_3=1}$$

With a method to represent a design point defined, the first population consisting of N_p design points (or candidate solutions) needs to be created. N_p is the size of the population. This means that N_p strings need to be created. In some cases, the designer already knows some good usable designs for the system. These can be used as seed designs to generate the required number of designs for the population using some random process. Otherwise, the initial population can be generated randomly via the use of a random number generator. Back to the example of three design variables $\mathbf{x} = [x_1, x_2, x_3]^{\mathrm{T}}$, N_p number of random variables of $C = 12$ digits can be generated. Let us say one of them is 803590043721. A rule can be set to convert the random number to a string of 0 and 1. The rule can be, for example, converting any value between zero and four to "0," and between five and nine to "1." Following this rule, the random number is converted to a string 100110000100, representing a design point of $x_1 = 9$ (decoded from string: 1001), $x_2 = 8$ (string: 1000), and $x_3 = 4$ (string: 0100).

Once a design point is created, its fitness function is evaluated. The fitness function defines the relative importance of a design. A higher fitness value implies a better design. The fitness function may be defined in several different ways. One commonly employed fitness function is

$$F_i = f_{\max} - f_i \tag{17.88}$$

where $f_{\max}$ is the maximum objective function value obtained by evaluating at each design point of the current population, and f_i and F_i are the objective function value and fitness function value of the ith design point, respectively.

17.7.1.3 Selection

The basic idea of a genetic algorithm is to generate a new set of designs (population) from the current set such that the average fitness of the population is improved, which is called a reproduction or selection process. Parents are selected according to their fitness; for example, each individual is selected with a probability proportional to its fitness value, say 50%, meaning that only half of the population with the best fitness functions is selected as parents to breed the next generation by undergoing genetic operations, to be discussed next. By doing so, weak solutions are eliminated and better solutions survive to form the next generation. The process is continued until a stopping criterion is satisfied or the number of iterations exceeds a specified limit.

17.7.1.4 Reproduction Process and Genetic Operations

There are many different strategies to implement the reproduction process; usually a new population (children) is created by applying recombination and mutation to the selected individuals (parents). Recombination creates one or two new individuals by swapping (crossing over) the genome of a parent with another. A recombined individual is then mutated by changing a single element (genome) to create a new individual. Crossover and mutation are the two major genetic operations commonly employed in the genetic algorithms.

Crossover is the process of combining or mixing two different designs (chromosomes) into the population. Although there are many methods for performing crossover, the most common ones are the

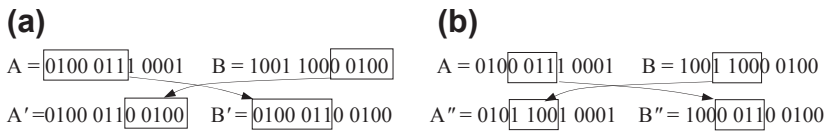


FIGURE 17.17

Crossover operations: (a) with one cut point and (b) with two cut points.

one-cut-point and two-cut-point methods. A cut point is a position on the genetic string. In the one-cut-point method, a position on the string is randomly selected that marks the point at which two parent design points (chromosomes) split. The resulting four halves are then exchanged to produce new designs (children). For example, string $A = 0100\ 0111\ 0001$, representing design $\mathbf{x}^A = (4, 7, 1)$ and string $B = 1001\ 1000\ 0100$ representing $\mathbf{x}^B = (9, 8, 4)$ are two design points of the current generation. If the cut point is randomly chosen at the seventh digit, as shown in Figure 17.17(a), the new strings become $A' = 0100\ 0110\ 0100$, in which the last five digits were replaced by those of string B, and $B' = 0100\ 0110\ 0100$, in which the first seven digits were replaced by those of string A. As a result, two new designs are created for the next generation, i.e., $\mathbf{x}^{A'} = (4, 6, 4)$ and $\mathbf{x}^{B'} = (4, 6, 4)$. These two new design points (children) $\mathbf{x}^{A'}$ and $\mathbf{x}^{B'}$ happen to be identical, which is less desirable.

Similarly, the two-cut-point method is illustrated in Figure 17.17(b), in which the two cut points are chosen as 3 and 7, respectively. The two new strings become $A'' = 0101\ 1001\ 0001$, in which the third to sixth digits were replaced by those of string B, and similarly $B'' = 1000\ 0110\ 0100$. As a result, two new designs are created for the next generation: $\mathbf{x}^{A''} = (5, 9, 1)$ and $\mathbf{x}^{B''} = (8, 3, 4)$.

Selecting how many or what percentage of chromosomes to crossover, and at what points the crossover operation occurs, is part of the heuristic nature of genetic algorithms. There are many different approaches, and most are based on random selections.

Mutation is another important operator that safeguards the process from a complete premature loss of valuable genetic material during crossover. In terms of a binary string, this step corresponds to the selection of a few members of the population, determining a location on the strings at random, and switching the 0 to 1 or vice versa.

Similar to the crossover operator, the number of members selected for mutation is based on heuristics, and the selection of location on the string for mutation is based on a random process.

Let us select a design point as "1000 1110 1001" and select the 7th digit to mutate. The mutation operation involves replacing the current value of 1 at the seventh location with 0 as "1000 1100 1001."

For example, string $C = 1110\ 0111\ 0101$ represents design $\mathbf{x}^C = (14, 7, 5)$. If we choose the seventh digit to mutate, the new strings become $C' = 1110\ 0101\ 0101$. As a result, the new design is created for the next generation as $\mathbf{x}^{C'} = (14, 5, 5)$.

In numerical implementation, not all individuals in a population go through crossovers and mutations. Too many crossovers can result in a poorer performance of the algorithm because it may produce designs that are far away from the mating designs (designs of higher fitness value). The mutation, on the other hand, changes designs in the neighborhood of the current design; therefore, a larger amount of mutation may be allowed. Note also that the population size needs to be set to a reasonable number for each problem. It may be heuristically related to the number of design variables and the number of all possible designs determined by the number of allowable discrete values for each variable. Key parameters, such as the number of crossovers and mutations, can be adjusted to fine-tune

the performance of the algorithm. In general, the probability of crossover being applied is typically less than 0.1, and probability of mutation is between 0.6 and 0.9.

17.7.1.5 Solution Process

A typical solution process of a genetic algorithm is illustrated in Figure 17.18. The initial population is usually generated randomly in accordance with the encoding scheme. Once a population is created, fitness function is assigned or evaluated to individuals in the population, for example, using Eq. 17.88. Parents are then selected according to their fitness. Genetic operations, including crossover and mutation, are performed to create children. The fitness of the new population is evaluated and the process is repeated until stopping criteria are met. The stopping criteria include (a) the improvement for the best objective (or pseudo-objective) function value is less than a small tolerance ε_0 for the last n consecutive iterations (n is chosen by the users), or (b) if the number of iterations exceeds a specified value.

17.7.2 SIMULATED ANNEALING

Simulated annealing is a stochastic approach that locates a good approximation to the global minimum of a function. The main advantage of SA is that it can be applied to a broad range of problems regardless of the conditions of differentiability, continuity, and convexity that are normally required in conventional optimization methods. Given a long enough time to run, an algorithm based on this concept finds global minima for an optimization problem that consists of continuous, discrete, or integer variables with linear or nonlinear functions that may not be differentiable.

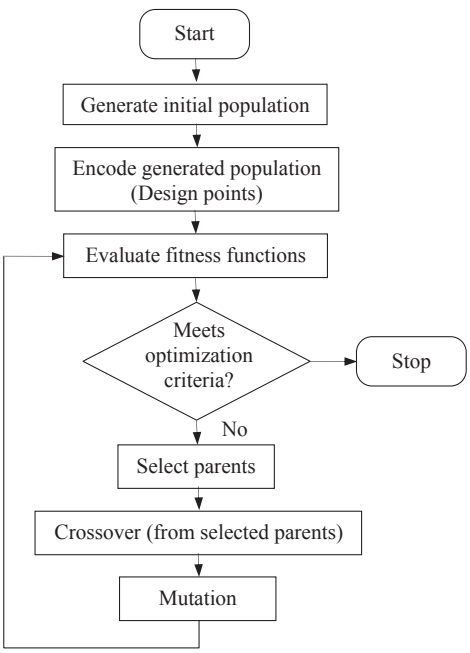


FIGURE 17.18

A typical solution process of a genetic algorithm.

The name of the approach comes from the annealing process in metallurgy. This process involves heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. Simulated annealing emulates the physical process of annealing and was originally proposed in the domain of statistical mechanics as a means of modeling the natural process of solidification and formation of crystals. During the cooling process, it is assumed that thermal equilibrium (or quasi-equilibrium) conditions are maintained. The cooling process ends when the material reaches a state of minimum energy, which, in principle, corresponds with a perfect crystal.

17.7.2.1 Basic Concept

The basic idea for implementation of this analogy to the annealing process is to generate random points in the neighborhood of the current best design point and evaluate the problem functions there. If the objective function value is smaller than its current best value, the design point is accepted and the best function value is updated. If the function value is higher than the best value known thus far, the point is sometimes accepted and sometimes rejected. Nonimproving (inferior) solutions are accepted in the hope of escaping a local optimum in search of the global optimum. The probability of accepting non-improving solutions depends on a "temperature" parameter, which is typically nonincreasing with each iteration. One commonly employed acceptance criterion is stated as

$$P(\mathbf{x}') = \begin{cases} e^{-\frac{f(\mathbf{x}')-f(\mathbf{x})}{T^k}}, & \text{if } f(\mathbf{x}') - f(\mathbf{x}) \geq 0 \\ 1, & \text{if } f(\mathbf{x}') - f(\mathbf{x}) < 0 \end{cases} \tag{17.89}$$

where $P(\mathbf{x}')$ is the probability of accepting the new design $\mathbf{x}'$ randomly generated in the neighborhood of $\mathbf{x}$, $\mathbf{x}$ is the current best point (of iteration k), and T^k is the temperature parameter at the kth iteration, such that

$$T^k > 0, \quad \text{for all } k; \quad \text{and} \quad \lim_{k \to \infty} T^k = 0 \tag{17.90}$$

where temperature parameter T^k is positive, and is decreasing gradually to zero when the number of iterations k reaches a prescribed level. In implementation, k is not approaching infinite but a sufficiently large number. Also, as shown in Eq. 17.89, the probability of accepting an inferior design depends on the temperature parameter T^k. At earlier iterations, T^k is relatively large; hence, the probability of accepting an inferior design is higher, providing a means to escape the local minimum by allowing so-called hill-climbing moves. As the temperature parameter T^k is decreased in later iterations, hill-climbing moves occur less frequently, and the algorithm offers a better chance to converge to a global optimal. Hill-climbing is one of the key features of the simulated annealing method.

As shown in Eq. 17.90, the algorithm demands a gradual reduction of the temperature as the simulation proceeds. The algorithm starts initially with T^k set to a high value, and then it is decreased at each step following some annealing schedule—which may be specified by the user, or set by a parameter, such as

$$T^{k+1} = rT^k, r < 1 \tag{17.91}$$

where T^{k+1} is the temperature for the next iteration $k + 1$.

17.7.2.2 Solution Process

We start the solution process by choosing an initial temperature T^0 and a feasible trial point $\mathbf{x}^0$. Compute objective function $f(\mathbf{x}^0)$. Select an integer L (which sets a limit on the number of iterations), and a parameter $r < 1$.

At the kth iteration, we then generate a new point $\mathbf{x}'$ randomly in a neighborhood of the current point $\mathbf{x}^k$. If the point is infeasible, generate another random point until feasibility is satisfied. Check if the new point $\mathbf{x}'$ is acceptable. If $f(\mathbf{x}') < f(\mathbf{x}^k)$, then accept $\mathbf{x}'$ as the new best point, set $\mathbf{x}^{k+1} = \mathbf{x}'$, and continue the iteration. If $f(\mathbf{x}') \geq f(\mathbf{x}^k)$, then calculate $P(\mathbf{x}')$ using Eq. 17.89. In the meantime, we generate a random number z uniformly distributed in [0, 1]. If $P(\mathbf{x}') > z$, then accept $\mathbf{x}'$ as the new best point and set $\mathbf{x}^{k+1} = \mathbf{x}'$ to continue the iteration. If not, we generate another new point $\mathbf{x}''$ randomly, and repeat the same steps until a new best point is found. We repeat the iteration, until k reaches the prescribed maximum number of iterations L.

17.8 PRACTICAL ENGINEERING PROBLEMS

We have introduced numerous optimization problems and their solution techniques. So far, all example problems we presented assume explicit expressions of objective and constraint functions in design variables. In practice, there are only a very limited number of engineering problems, such as a cantilever beam or two-bar truss systems, that are simple enough to allow for explicit mathematical expressions in relating functions with design variables. In general, explicit expressions of functions in design variables are not available. In these cases, software tools such as MATLAB are not applicable.

Solving design optimization problems that involve function evaluations of physical problems that do not have explicit function expressions in terms of design variables deserve our attention because they are very common to design engineers in practice. Some of these problems require substantial computation time for function evaluations. In either case, we must rely on commercial software to solve these problems.

Commercial CAD or CAE tools with optimization capabilities offer viable capabilities for solving practical engineering problems. They are usually the first options that designers seek for solutions. One key factor to consider in using CAD tools for optimization is design parameterization using geometric dimensions. The part or assembly must be fully parameterized so that the solid models can be updated in accordance with design changes during optimization iterations. We briefly discuss design parameterization from optimization perspective. For more details, readers are referred to Chapter 5 Design Parameterization. We also provide a brief overview on some of the popular commercial software in Section 17.9 and offer tutorial examples for further illustrations in Section 17.11.

In some situations, commercial software may not offer sufficient or adequate capabilities that support design needs. For one, commercial tools may not offer function evaluations for the specific physical problems at hand. For example, if fatigue life is to be maximized for a load-carrying component, the commercial tool employed must provide adequate fatigue life computation capability. In addition, the tool must allow users to include fatigue life as an objective or constraint function for defining the optimization problem, and underline optimization algorithm must be fully integrated with the fatigue life computation capability. Another situation could be that the optimization capabilities offer in the commercial tool employs solution techniques that require many function evaluations, such as genetic algorithm. For large-scale problems, carrying out many analyses for function

evaluations may not be feasible computation-wise. One may face a situation that multiple commercial codes need to be integrated to solve the problems at hand. We discuss tool integration for optimization in Section 17.8.1, which provides readers with the basic ideas and a sample case for such a need.

Finally in many cases, a true minimum is not necessarily sought; instead, an improved design is sufficient, especially for large-scale problems that require days for function evaluations. In this situation, batch mode optimization that takes several design iterations to converge may not be feasible. We present an interactive design process in Section 17.8.2 that significantly reduces the number of function evaluations and design iterations. This interactive process supports solving large-scale problems often in just one of two design iterations.

17.8.1 TOOL INTEGRATION FOR DESIGN OPTIMIZATION

Three major tools are critical for support of modeling (including design parameterization), analysis, and optimization. Modeling tool provides designers the capability of creating product design model, either part (in CAD or FEA tools) or assembly. Analysis tools support function evaluations, depending on the kinds of physical problems being solved. The physical problems can be single disciplinary, which involve a single engineering discipline, such as structural analysis with function evaluations for stress, displacement, buckling load factor, and so forth. On the other hand, the physical problems can be multidisciplinary, in which two or more engineering disciplines are involved, such as motion analysis of a mechanism (for reaction force calculation), structural analysis for load-carrying components of the mechanism, machining simulation (for machining time calculation and manufacturing cost estimate), and so forth. Finally, optimization tools that offer desired optimization algorithms for searching for optimal design are essential. In many situations, gradient calculations (also called design sensitivity analysis) play an important role in design optimization. This is because, in general, gradient-based optimization techniques that require much less function evaluations compared to the non-gradient approaches are the only viable approach to support design problems involved large-scale physical models. Moreover, accurate gradient information facilitates the search for optimal solutions, and often requires fewer design iterations. More about gradient calculations is discussed in Chapter 18.

In the following, we present a case of tool integration for CAD-based mechanism optimization, in which kinematic and dynamic analysis is required for function evaluations. This integrated system has been applied to support the suspension design of a high-mobility multipurpose wheeled vehicle (HMMWV), as mentioned in Sections 1.6, 5.6.2, and 8.6.2.

In this integrated system, commercial codes are first sought to support modeling, analysis, and optimization. In addition to the commercial codes, a number of software modules need to be implemented to support the tool integration, such as interface modules for data retrieval and model update modules for updating CAD and simulation models in accordance with design changes. The overall flowchart of the software system that supports CAD-based mechanism optimization is illustrated in Figure 17.19. The system consists of Pro/ENGINEER and SolidWorks for product model representation, Dynamic Analysis and Design System (DADS; Haug and Smith, 1990) for kinematic and dynamic analysis of mechanical systems including ground vehicles, and Design Optimization Tool (DOT; www.vrand.com/dot.html) for a gradient-based design optimization. In this case, the overall finite difference method has been adopted to support gradient calculations.

In this system, engineers will create parts and assemblies of the product in a CAD tool. The solid model will be parameterized by properly generating part features and assembly constraints, as

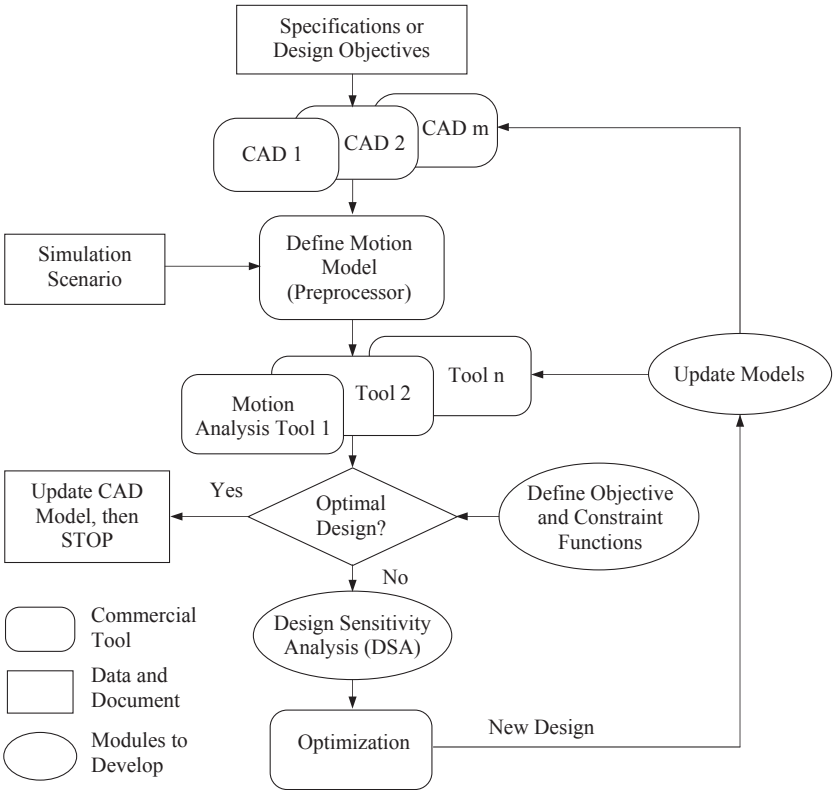


FIGURE 17.19

Overall flow of the CAD-based mechanism optimization.

well as relating geometric dimensions to capture the design intents, following those discussed in Chapter 5. Independent geometric dimensions of significant influence on the motion characteristics of the mechanical system are chosen as design variables. Consequently these variables help engineers achieve the design objectives more effectively.

The preprocessor supports design engineers in defining a complete motion model derived from the CAD solid model. Key steps include assigning a body local coordinate system (usually the default coordinate system in CAD), defining connections (or joints) between bodies, specifying initial conditions, and creating loads and drivers for dynamic and kinematic analyses. More about motion model creation and simulation can be found in Chapter 8 Motion Analysis.

Design sensitivity analysis (DSA) calculates gradients of motion performance measures of the mechanical system with respect to dimension design variables in CAD. This is critical for optimization. The gradient information provides engineers with valuable information for making design decisions. At the same time, it supports gradient-based optimization algorithms in searching for an optimal design. The gradient information and performance measure values are provided to the optimization algorithms in order to find improved designs during optimization iterations. In general, an

analytical DSA method for gradient calculations is desirable, in which the derivative of a motion performance ψ with respect to CAD design variables $\mathbf{x}$ can be expressed as follows:

$$\frac{\partial \psi(\mathbf{d},\mathbf{c})}{\partial \mathbf{x}} = \frac{\partial \psi(\mathbf{d},\mathbf{c})}{\partial \mathbf{m}}\frac{\partial \mathbf{m}}{\partial \mathbf{d}} + \frac{\partial \psi(\mathbf{d},\mathbf{c})}{\partial \mathbf{p}}\frac{\partial \mathbf{p}}{\partial \mathbf{d}} + \frac{\partial \psi(\mathbf{d},\mathbf{c})}{\partial \mathbf{c}} \tag{17.92}$$

where $\mathbf{x} = [\mathbf{d}^{\mathrm{T}}, \mathbf{c}^{\mathrm{T}}]$ is the vector of design variables; $\mathbf{d}$ is the vector of dimension design variables captured in CAD solid models; $\mathbf{c}$ is a vector of physical parameters included in a load or a driver, such as the spring constant of a spring force created in the analysis model; $\mathbf{m}$ is the vector of mass property design variables, including mass center, total mass, and moment of inertia; and $\mathbf{p}$ is the vector of joint position design variables. In many situations, analytical methods for gradient calculations are not available; the overall finite difference method is an acceptable alternative, in which the derivative of a motion performance ψ with respect to CAD design variables $\mathbf{x}$ can be expressed as follows:

$$\frac{\partial \psi(\mathbf{x})}{\partial x_i} \approx \frac{\Delta\psi(\mathbf{x})}{\Delta x_i} = \frac{\psi(\mathbf{x} + \Delta x_i) - \psi(\mathbf{x})}{\Delta x_i} \tag{17.93}$$

where $\psi(\mathbf{x})$ is a dynamic performance of the mechanical system at the current design, and $\psi(\mathbf{x} + \Delta x_i)$ is the performance at the perturbed design with a design perturbation Δx_i for the ith design variable. Note that the design perturbation Δx_i is usually very small.

The motion model must be updated after a new design is determined in the optimization iterations. Mass properties and joint locations of the new design must be recalculated according to the new design variable values. The new properties and locations will replace the existing values in the input data file or binary database of the motion model for motion analysis in the next design iteration. Note that the definition of the motion model is assumed to be unchanged in design iterations; for example, no new body or joint can be added, driving conditions cannot be altered, and the same road condition must be kept during design iterations. Mass properties, joint locations, and physical parameters that define forces or torque (e.g., spring constant) are allowed to change during design iterations.

A simple slider crank example shown in Figure 17.20(a) is presented to demonstrate the feasibility of the integrated system. Schematically, the mechanism is a standard 4-bar linkage, as illustrated in Figure 17.20(b). Moreover, geometric features in the crankshaft and connecting rod have been created with proper dimensions and references such that when their lengths are changed, the entire parts vary accordingly. At the assembly level, when either of the two length dimensions $d2{:}0$ or $d3{:}2$ is changed,

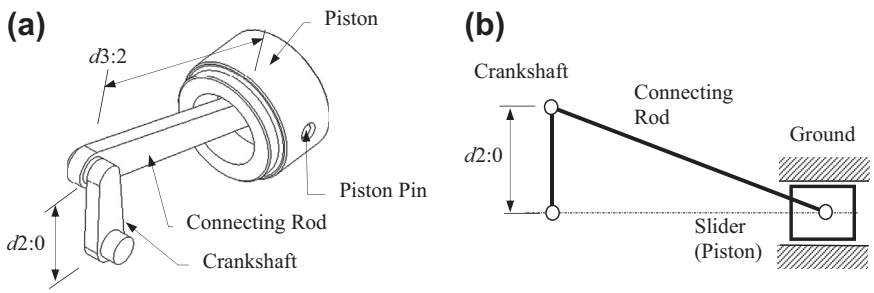


FIGURE 17.20

A slider-crank mechanism: (a) CAD solid model and (b) schematic view.

the change is propagated to the affected parts. The remaining parts are kept unchanged, and the entire assembly is kept intact, as illustrated in Figure 5.7 of Chapter 5.

The CAD and motion models are created in SolidWorks and SolidWorks Motion, respectively. The mechanism is driven by a constant torque of 10 in-lb. applied to the crankshaft for 5 seconds. Note that friction is assumed to be nonexistent in any joint. The optimization problem is formulated as follows:

$$\text{Minimize:} \quad \phi(\mathbf{x}) \tag{17.94a}$$

$$\text{Subject to:} \quad \psi_1(\mathbf{x}) \leq 9000 \text{ lb} \tag{17.94b}$$

$$\psi_1(\mathbf{x}) \leq 9000 \text{ lb} \tag{17.94c}$$

$$1.0 \leq x_1 \leq 4.0 \text{ in} \tag{17.94d}$$

$$0.2 \leq x_2 \leq 1.0 \text{ in} \tag{17.94e}$$

$$5.0 \leq x_3 \leq 8.0 \text{ in} \tag{17.94f}$$

where the objective function $\phi(\mathbf{x})$ is the total volume of the mechanism; $\psi_1(\mathbf{x})$ and $\psi_2(\mathbf{x})$ are the maximum reaction forces in magnitude at the joints between crank and bearing, and between crank and rod, respectively, during a 5-sec. simulation period; and x_1, x_2, and x_3 are design variables specifying the crankshaft length (d2:0), crankshaft width, and rod length (d3:2), respectively. At the initial design, we have $x_1 = 3$ in, $x_2 = 0.5$ in, and $x_3 = 8$ in, as shown in Figure 17.21(a). The maximum reaction forces are 10,130 and 9,670 lb, respectively; therefore, the initial design is infeasible. The optimization took 15 iterations to converge using the modified feasible direction (MFD) algorithm (Vanderplaats, 2005) in DOT. At the optimum, the overall volume of the mechanism is reduced by 11%, both performance constraints are satisfied, and two out of the three design variables reached their respective lower bounds, as listed in Table 17.1. The optimized mechanism is shown in Figure 17.21(b). Note that all three design variables are reduced to achieve an optimal design because the design directions that minimize the total volume and reduce reaction forces between joints (due to mass inertia) happen to be consistent.

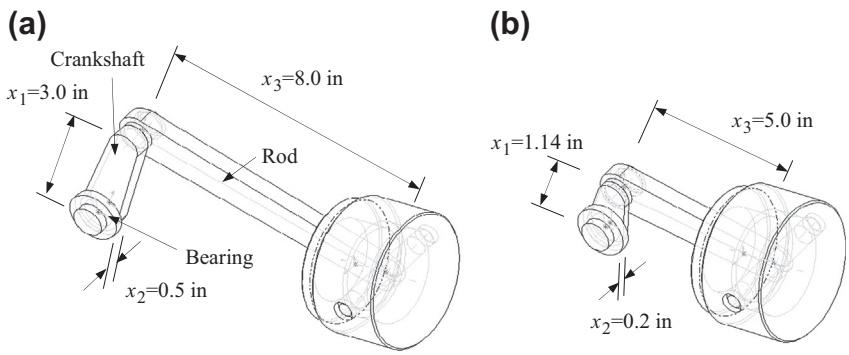


FIGURE 17.21

Design optimization of the slider-crank mechanism: (a) initial design with design variables and (b) optimal design (Chang and Joo, 2006).

Table 17.1 Design Optimization of the Slider-Crank Mechanism (Chang and Joo, 2006)

Measure	Initial Design	Optimal Design	% Change
$\phi(\mathbf{x})$	28.42 in^3	25.21 in^3	−11.3
$\psi_1(\mathbf{x})$	10,130 lb	9015 lb	−9.9
$\psi_2(\mathbf{x})$	9670 lb	8854 lb	−8.4
x_1	3 in	1.14 in	−62.0
x_2	0.5 in	0.20 in	−60.0
x_3	8.0 in	5.0 in	−37.5

17.8.2 INTERACTIVE DESIGN PROCESS

For practical design problems, a true optimal is not necessarily sought. Instead, an improved design that eliminates known problems or deficiencies is often sufficient. Also, as mentioned earlier, function evaluations require substantial computation time for large-scale problems. Therefore, even using the gradient-based solution techniques, the computation may take too long to converge to an optimal solution.

In addition to batch-mode design optimization, an interactive approach that reduces the number of function evaluations and design iterations is desired, especially for large-scale problems. In this subsection, we discuss one such approach called three-step design process, involving sensitivity display, what-if study, and trade-off analysis. The interactive design approach mainly supports the designer in better understanding the behavior of the design problem at the current design and suggests design changes that effectively improve the product performance in one or two design iterations.

17.8.2.1 Sensitivity Display

The derivatives (or gradients) of functions with respect to design variables are called design sensitivity coefficients or gradients. These coefficients can be calculated, for example, using the overall finite difference method mentioned in Section 17.8.1. Once calculated, the coefficients can be shown in a matrix form, such as in a spreadsheet, as shown in Figure 17.22. Individual rows in the matrix display numerical numbers of gradients for a specific function (e.g., von Mises stress in Row 3 in Figure 17.22) with respect to all design variables; that is, $\partial\psi_i/\partial x_j$, $j = 1$, number of design variables (which is three in the matrix of Figure 17.22). Each row of the matrix shows the influence of design variables to the specific function (either objective or constraint). On the other hand, individual columns in the matrix display gradients of all functions with respect to a single design variable; that is, $\partial\psi_i/\partial x_j$, $i = 1$, number of functions. Each column shows the influence of the design variable to all functions.

The design sensitivity matrix contains valuable information for the designer to understand product behavior so as to make appropriate design changes. For example, if the von Mises stress is greater than the allowable stress, then the data in Row 3 show a plausible design direction that reduces the magnitude of the stress measure effectively. In the bar chart of the Row 3 data shown in the lower left of Figure 17.22, increasing the first design variable reduces stress significantly because the gradient is negative with a largest magnitude. On the other hand, decreasing the second design variable reduces the stress because the gradient of the design variable is positive. Changing the third

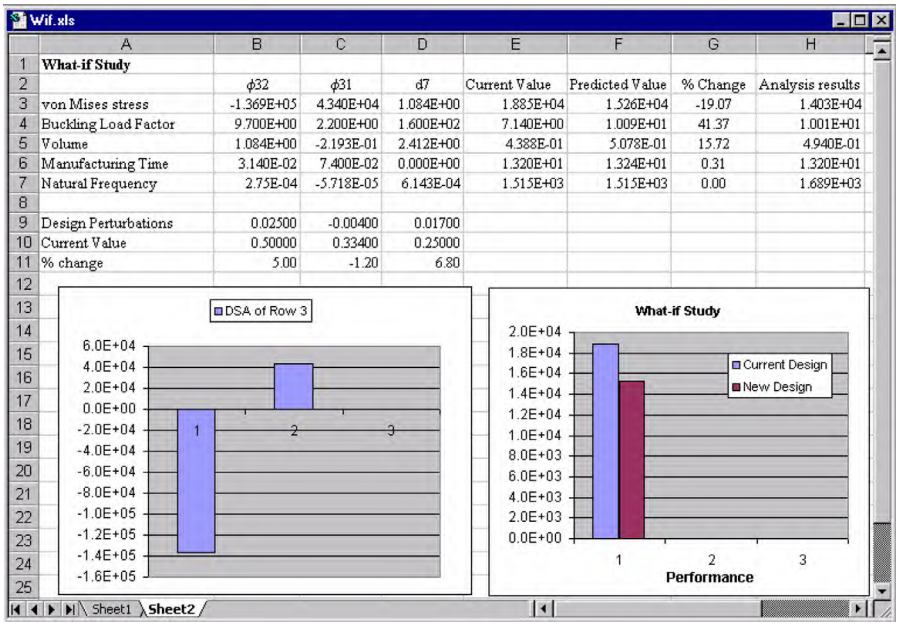


FIGURE 17.22

Spreadsheet showing sensitivity matrix and bar charts for better visualization.

design variable does not impact the stress because its gradient is very small. In fact, data in each row show the steepest ascent direction for the respective function (in increasing its function value). Reversing the direction gives the steepest descent direction for design improvement. By reviewing row or column data of the sensitivity matrix in bar charts, designers gain knowledge in the behavior of the design problem and possibly come up with a design that improves respective functions. Once a design change is determined, the designer can carry out a what-if study that calculates the function values at the new design using first-order approximations without going over expensive analyses for function evaluations.

17.8.2.2 What-if Study

A what-if study, also called a parametric study, offers a quick way for the designer to find out "What will happen if I change a design variable this small amount?" In a what-if study, the function values at the new design are approximated by the first-order Taylor's series expansion:

$$\psi_i(\mathbf{x} + \delta x_j) \approx \psi_i(\mathbf{x}) + \frac{\partial \psi_i}{\partial x_j}\delta x_j \tag{17.95}$$

where ψ_i is the ith function of the design problem, $\partial\psi_i/\partial x_j$ is the design sensitivity coefficient of the ith function with respect to the jth design variable, and δx_j is the design perturbation of the jth design variable. The what-if study gives quick first-order approximation for product performance measures at the perturbed design without going through a new analysis.

In general, a design perturbation $\delta\mathbf{x}$ is provided by the designer, calculated along the steepest descent direction of a function with a given step size, or computed in the direction found by the trade-off determination to be discussed next, with a step size α; that is, $\delta\mathbf{x} = \alpha\mathbf{n}$. The what-if study results and the current function values can be displayed in a spreadsheet or bar charts, such as that shown in Figure 17.22 (lower right). These kinds of displays allow the designer to instantly identify the predicted performance values at the perturbed design and compare them with those at the current design.

Once a better design is obtained from the what-if study (in approximation), the designer may commit to the changes by updating the CAD and analysis models, then carrying out analyses for the updated model to confirm that the design is indeed better. This completes one design iteration in an interactive fashion. Note that to ensure reasonably accurate function predictions using Eq. 17.95, the step size α must be small so that the perturbation $\frac{\partial \psi_i}{\partial \mathbf{x}}\delta\mathbf{x}$ is, as a rule of thumb, less than 10% of the function value $\psi_i(\mathbf{x})$.

17.8.2.3 Trade-off Determination

In many situations, functions are in conflict resulting from a design change. A design may bring an infeasible design into a feasible region by reducing constraint violations; in the meantime, the same change could increase the objective function value that is undesirable. Therefore, very often in the design process, the designer must carry out design trade-offs between objective and constraint functions.

The design trade-off analysis method presented in this section assists the designer in finding the most appropriate design search direction of the optimization problem formulated in Eq. 17.3, using four possible options: (1) reduce cost (objective function value), (2) correct constraint neglecting cost, (3) correct constraint with a constant cost, and (4) correct constraint with a cost increment. As a general rule of thumb, the first algorithm, reduce cost, can be chosen when the design is feasible. When the current design is infeasible, among the other three algorithms, generally one may start with the third option, correct constraint with a constant cost. If the design remains infeasible, the fourth option, correct constraint with a cost increment of, say, 10%, may be chosen. If a feasible design is still not found, the second algorithm, correct constraint neglecting cost, can be selected. A QP subproblem, discussed in Section 17.6.4, is formulated to find the search direction numerically corresponding to the option selected.

The QP subproblem for the first option (cost reduction) can be formulated as

$$\text{Minimize:} \quad \bar{f} = \mathbf{c}^{\mathrm{T}}\mathbf{d} + \frac{1}{2}\mathbf{d}^{\mathrm{T}}\mathbf{d} \tag{17.96a}$$

$$\text{Subject to:} \quad \mathbf{A}^{\mathrm{T}}\mathbf{d} \leq \mathbf{b} \tag{17.96b}$$

$$\mathbf{N}^{\mathrm{T}}\mathbf{d} = \mathbf{e} \tag{17.96c}$$

which is identical to those of Eq. 17.74 in Section 17.6.4.

For the second option (constraint correction neglecting cost), the QP subproblem is formulated as

$$\text{Minimize:} \quad \bar{f} = \frac{1}{2}\mathbf{d}^{\mathrm{T}}\mathbf{d} \tag{17.97a}$$

$$\text{Subject to:} \quad \mathbf{A}^{\mathrm{T}}\mathbf{d} \leq \mathbf{b} \tag{17.97b}$$

$$\mathbf{N}^{\mathrm{T}}\mathbf{d} = \mathbf{e} \tag{17.97c}$$

Note that Eq. 17.97 is similar to Eq. 17.96, except that the first term of the objective function in Eq. 17.96a is deleted in Eq. 17.97a because the cost (objective function) is neglected.

For the third option (constraint correction with a constant cost), the QP subproblem is the same as Eq. 17.97, with an additional constraint $\mathbf{c}^{\mathrm{T}}\mathbf{d} \leq 0$ (implying no cost increment). For the last option (constraint correction with a specified cost), the QP subproblem is the same as Eq. 17.97, with an additional constraint $\mathbf{c}^{\mathrm{T}}\mathbf{d} \leq \Delta$, where Δ is the specified cost increment (implying the cost cannot increase more than Δ). The QP subproblems can be solved using a QP solver, such as MATLAB.

After a search direction $\mathbf{d}$ is found by solving the QP subproblem for the respective option, it is normalized to the vector $\mathbf{n} = \mathbf{d}/||\mathbf{d}||$. Thereafter, a number of step sizes α_s can be used to perturb the design. Objective and constraint function values, represented as ψ_i, at a perturbed design $\mathbf{x} + \alpha_i\mathbf{n}$ can be approximated by carrying out a what-if study. Once a satisfactory design is identified after trying out different step sizes in an approximation sense, the design model can be updated to the new design for the next design iteration.

A particular advantage of the interactive design approach is that the designer can choose the proper option to perform design trade-offs and carry out what-if studies efficiently, instead of depending on design optimization algorithms to find a proper design by carrying out line searches using several function evaluations, which is expensive for large-scale problems. The result of the interactive design process can be a near-optimal design that is reliable. A case study is presented in Section 17.10.1 using a tracked-vehicle roadwheel example to further illustrate the interactive design process.

17.9 OPTIMIZATION SOFTWARE

There are numerous commercial optimization software tools that support engineering design. We have seen throughout this chapter the use of MATLAB to carry out optimization for both mathematical and engineering problems. In using MATLAB, the objective and constraint functions must be explicitly expressed in terms of design variables. Although easy to use, there are only a limited number of engineering problems that are simple enough to allow for explicit mathematical expressions in relating functions with design variables.

In this section, we provide a brief review on commercial software tools by which engineering optimization problems beyond simple beams and trusses can be solved. We categorize them into optimization in CAD and optimization in FEA that offer general optimization capabilities. We also include special-purpose codes that are tailored for solving specific types of optimization problems.

17.9.1 OPTIMIZATION IN CAD

In most cases, optimization capability is implemented as a software module in commercial software that offers more engineering capabilities than just optimization. Some of them are embedded in CAD software, such as Pro/ENGINEER and SolidWorks, and more are available in CAE software,

including ANSYS (www.ansys.com), MSC/NASTRAN (www.mscsoftware.com/product/msc-nastran), and LS-OPT (www.lstc.com/products/ls-opt).

Pro/MECHANICA Structure (or Mechanica) embedded and fully integrated in Pro/ENGINEER supports optimization for structural problems, in which geometric dimensions and physical parameters, such as material properties, of a part or assembly can be included as design variables. Functions, such as stress, displacement, buckling load factors, and weight, can be incorporated as objective and constraint functions. Mechanica employs a standard gradient-based optimization technique in searching for optimal solutions. If the part or assembly in Pro/ENGINEER is fully parameterized, optimization can be carried out fully automatically.

SolidWorks Simulation also supports engineering optimization. Similar to Mechanica, Simulation supports optimization for structural problems. Instead of using the gradient-based solution technique, Simulation employs a generative method to search for a solution, as discussed in Section 17.5.1. As a result, more FEAs are needed, yielding only a near-optimal solution.

In addition to Pro/ENGINEER and SolidWorks, commercial CAD software tools, such as CATIA V5 and NX CAE 8.5, support design optimization for structural problems.

The key advantages of solving optimization problems using CAD are twofold. First, CAD supports parametric modeling. Hence, design variables are defined by choosing part dimensions. As long as the part or assembly is fully parameterized, optimization can be carried out fully automatically in most cases. Second, the FEA and optimization modules are fully integrated with CAD. Therefore, the design capabilities, including choosing objective and constraint functions, as well as selecting design variables are straightforward and easy to use. Tutorial examples are provided in Section 17.11 and Projects P5 and S5 to illustrate details in using Mechanica and Simulation for structural optimization.

17.9.2 OPTIMIZATION IN FEA

Another group of optimization codes are embedded in commercial FEA software, including MSC/NASTRAN, ANSYS DesignXplore (www.ansys.com), GENESIS (www.vrand.com/Genesis.html), and OptiStruct (www.altairhyperworks.com).

MSC/NASTRAN, developed by MSC Software Corporation in Newport Beach, California, is one of the most popular FEA software in industry. The software offers a set of most complete CAE capabilities, including structural, thermal, fluid, fatigue, dynamics, and more. Its solution 200: Design Optimization and Sensitivity Analysis supports sensitivity analysis and gradient-based optimization for three major types of structural design problems: sizing, shape, and topology optimization. With response functions and constraints supported across multiple disciplines, users do not have to perform multiple optimization runs for each discipline. It is possible to combine all these disciplines into a single run, so that users gain efficiency and obtain better designs.

ANSYS is another powerful and popular CAE software. DesignXplorer of ANSYS offers capabilities for designers to explore design alternatives, including optimization. DesignXplore employs a generative approach similar to that of SolidWorks Simulation, in which a design space is subdivided to create a series of simulation experiments for exploring better designs. With the simulation results obtained from simulation experiments, DesignXplore employs response surface technologies that interpolate between the data points in multidimensional design space. The

interpolated results can be visualized as a 2-D or 3-D description of the relationships between design variables and performance functions. Optimal design is then searched on the response surface.

GENESIS, developed by Vanderplaats Research & Development, is a fully integrated finite element analysis and design optimization software package. Analysis is based on the finite element method for static, normal modes, direct and modal frequency analysis, random response analysis, heat transfer, and system buckling calculations. Design is based on the gradient-based solution techniques—more specifically the SLP, SQP, and feasible direction methods. These approximate problems generated using analysis and sensitivity information, are used for the optimization, which is performed by DOT (or BIGDOT) optimizers. When the optimum of the approximate problem has been found, a new finite element analysis is performed and the process is repeated until the solution has converged to an optimum. Many design options are available for users, including shape, sizing, and topology. More discussions on these different design optimizations are to be discussed in Chapter 18.

OptiStruct is one of the software modules of HyperWorks, which is the principal product offered by Altair Engineering—a product design and development, engineering software, and cloud computing software company in Detroit, Michigan. OptiStruct supports topology, sizing, and shape optimization to create better and more alternative design proposals leading to structurally sound and lightweight design. OptiStruct has been widely accepted by the automotive industry.

17.9.3 SPECIAL-PURPOSE CODES

In addition to optimization capability embedded in CAD and FEA software, there are optimization software tools that integrate commercial FEA and provide capabilities that support solving general design optimization problems. This software includes Tosca Structure (www.fe-design.com/en/products/tosca-structure), LS-OPT, and so forth.

Tosca Structure offers structural optimization by integrating with industry-standard FEA packages, including ABAQUS, ANSYS, and MSC/NASTRAN. It allows for rapid and reliable design of lightweight, stiff, and durable components and systems. Tosca offers topology and sizing design capabilities through two modules: Structure.topology and Structure.sizing, respectively.

LS-OPT is a graphical optimization tool that interfaces with LS-DYNA and allows the designer to structure the design process, explore the design space, and compute optimal designs according to specified constraints and objectives. The optimization capability in LS-OPT is based on response surface and design of experiments (similar to that of DesignXplore). The software allows the combination of multiple disciplines and/or cases for the improvement of a unique design.

17.10 CASE STUDIES

We present two case studies, both involving FEA for structural analysis. The first case, sizing optimization of roadwheel, aims to minimize volume and constrain deformation of a tracked-vehicle roadwheel by varying its thicknesses. The second case study optimizes the geometric shape of an engine connecting rod using a p-version FEA code.

17.10.1 SIZING OPTIMIZATION OF ROADWHEEL

In this case study, we present a sizing optimization for a tracked-vehicle roadwheel, in which thicknesses of shell finite elements are varying for better designs. Function evaluations are carried out using FEA, in which shell elements (instead of solid elements) are created from the surface geometric model created in a geometric modeling tool, MSC/PATRAN (www.mscsoftware.com/product/patran).

The roadwheels shown in Figures 17.23(a) and (b) are heavy-load-carrying components of the tracked vehicle suspension system. There are seven wheels on each side of the vehicle. The geometric model of the roadwheel is created in MSC/PATRAN as quadrilateral surface patches, as shown in Figure 17.23(c). The objective of this design problem is to minimize its volume with prescribed allowable deformation at the contact area, by varying its thicknesses.

17.10.1.1 Geometric Modeling and Design Parameterization

Due to symmetry, only half of the wheel is modeled for design and analysis. The outer diameter of the wheel is 25 in., with two cross-section thicknesses, 1.25 in. at rim section (dv5, dv6, and dv7) and 0.58 in. at hub section (dv1 to dv4), as shown in Figure 17.24(a). To model the wheel, 216 Coons patches and 432 triangular finite elements are created in the geometric and finite element models, respectively, using PATRAN.

Thicknesses of the wheel are defined as design variables, which are linked with surface patches along the circumferential direction of the wheel to maintain a symmetric design, as illustrated in Figures 17.24(b) and (c). Figure 17.24(d) shows patches along the inner edge of the wheel, in which patch thicknesses are linked together as design variable dv1. In a similar fashion, an additional six design variables are defined for the wheel, as shown in Figures 17.25(b) and (c). In the current design, we have dv1 = dv2 = dv3 = dv4 = 0.58 in., and dv5 = dv6 = dv7 = 1.25 in.

17.10.1.2 Analysis Model

ANSYS plate elements STIF63 are employed for finite element analysis. There are 432 triangular plate elements and 1650 degrees of freedom defined in the model, as shown in Figure 17.25(a). This wheel is made up of aluminum with modulus of elasticity, $E = 10.5 \times 10^6$ psi, shear modulus, $G = 3.947 \times 10^6$ psi, and Poisson's ratio, $\nu = 0.33$.

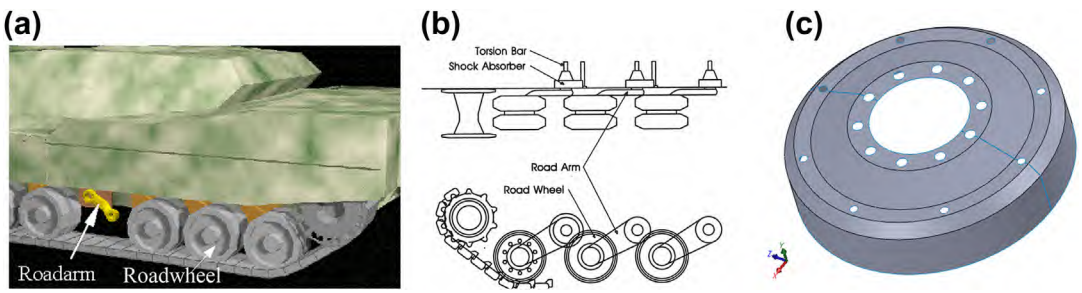


FIGURE 17.23

Tracked vehicle roadwheel. (a) Suspension showing roadarm and roadwheel. (b) Schematic view of the suspension, front and top views. (c) Geometric model of roadwheel in MSC/PATRAN (recreated in SolidWorks).

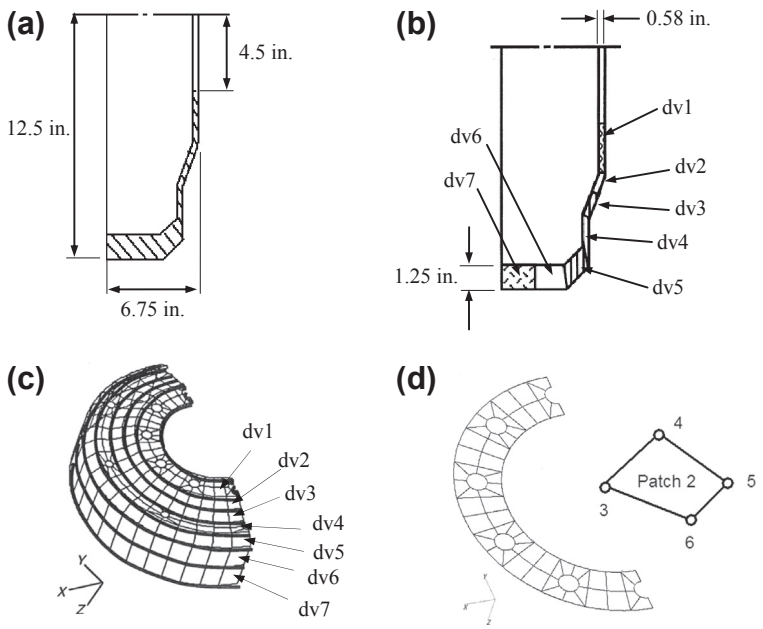


FIGURE 17.24

Roadwheel geometric model and design parameterization. (a) Half wheel section view with key dimensions. (b) The seven thickness design variables in section view. (c) Thicknesses of patches along the circumferential direction linked as design variables. (d) A closer view of the patches in the inner edge of the wheel.

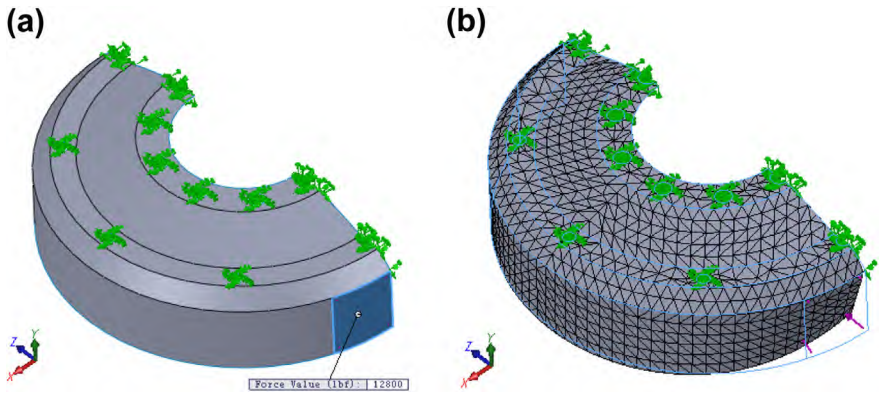


FIGURE 17.25

Finite element model of the roadwheel. (a) Boundary conditions, and loads applied. (b) Deformed shape obtained from FEA using ANSYS (Chang et al., 1992) (recreated in SolidWorks Simulation).

Table 17.2 Design Sensitivity Matrix

Performance	dv1	dv2	dv3	dv4	dv5	dv6	dv7
Displacement (in.)	−0.045865	−0.01553	−0.011015	−0.019515	−0.019420	−0.056306	−0.099343
Volume (in.^{3})	35.1301	26.2068	29.8408	34.8080	47.7519	93.8567	92.4915

The circumference of the six small holes in the hub area is fixed. Symmetric conditions are imposed at the cutoff section, and a distributed load of total 12,800 lb is applied to the six elements in the area where the wheel contacts the track. A deformed wheel shape, obtained from ANSYS analysis results, is displayed in PATRAN for result evaluation, as shown in Figure 17.25(b). From the analysis results, it is found that the maximum displacement occurs at the contact area, more specifically node 266, in the y-direction with magnitude 0.108 in. The volume of the wheel is 361.9 in^3.

17.10.1.3 Performance Measures

The maximum displacement (found at 266 in the y-direction) and wheel volume are defined as performance measures for the design problem.

17.10.1.4 Design Sensitivity Results and Display

In this case, gradients of the displacement and volume are calculated using the sensitivity analysis method based on the continuum formulation to be discussed in Chapter 18. Design sensitivity coefficients computed are listed in Table 17.2.

To display design sensitivity coefficients, both a PATRAN fringe plot and bar chart are utilized. Figure 17.26 shows the design sensitivity coefficients of the displacement performance measure. The

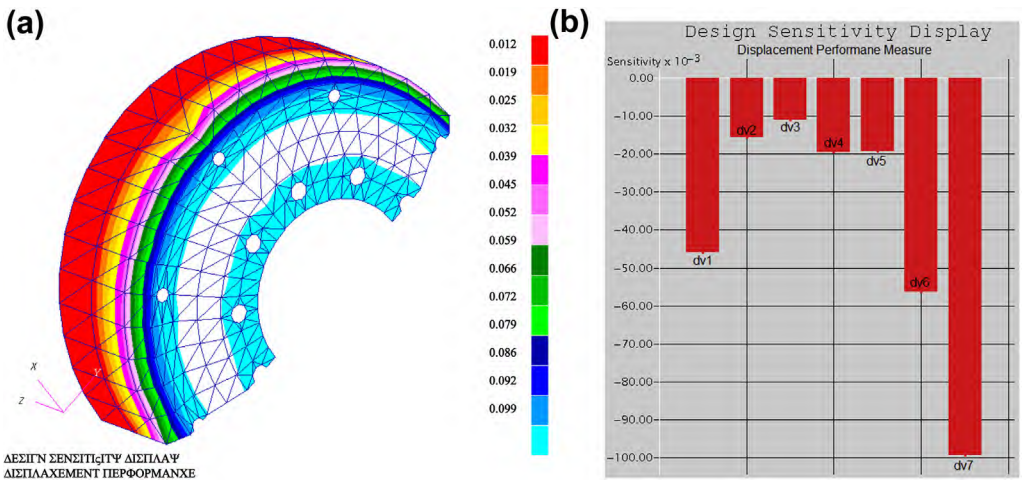


FIGURE 17.26

Design sensitivity of displacement performance measure: (a) fringe plot in PATRAN and (b) bar chart.

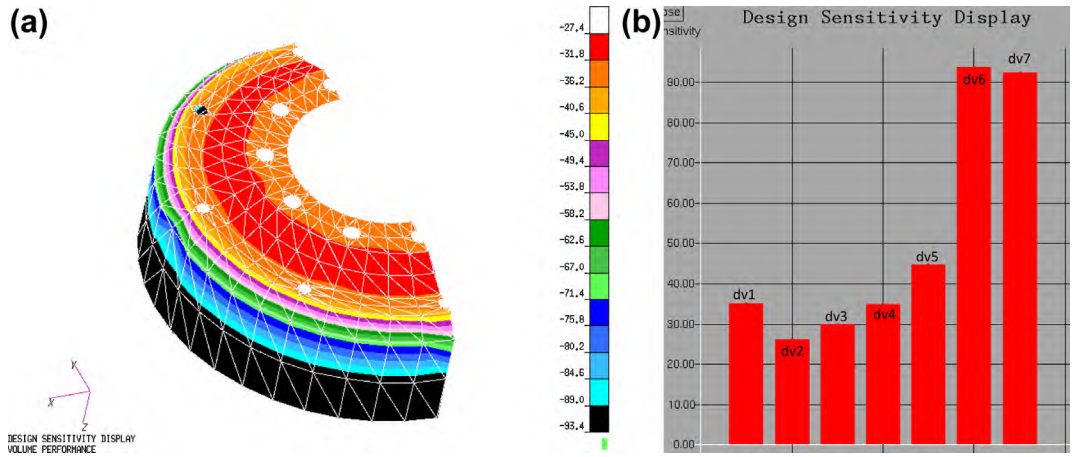


FIGURE 17.27

Design sensitivity of volume performance measure: (a) fringe plot in PATRAN and (b) bar chart.

design sensitivity plots clearly indicate that increasing the thickness at the outer edge of the wheel, that is, design variable dv7, reduces the displacement most significantly. As shown in the bar chart, a 1-in. increment of thickness at the outer edge of the wheel yields a 0.0993 in. reduction in the displacement. This influence decreases from the outer to inner edges of the wheel. At the inner edge of the wheel, the influence is increasing to about 40% of the maximum value.

Figure 17.27 shows the design sensitivity coefficients of the volume performance measure. The result shows that increasing thickness around outer edge of the wheel, dv6 and dv7, increases the volume performance measure most significantly. As shown in the bar chart, a 1-in. increment of thickness around the outer edge of the wheel yields 93.4 $in.^3$ and 92.5 $in.^3$ gain in wheel volume. The rate of influence decreases from the outer to the inner edges of the wheel.

17.10.1.5 What-if Study

A what-if study is carried out based on the steepest descent direction of the displacement performance measure with a step size of 0.1 in. The design direction shown in Table 17.3 suggests the most effective

Table 17.3 Design Direction in the Steepest Descent Direction

Design Variable	Current Value (in.)	New Value (in.)	% Change
Dv1	0.58	0.61596	6.20
Dv2	0.58	0.59218	2.10
Dv3	0.58	0.58864	1.49
Dv4	0.58	0.59530	2.64
Dv5	1.25	1.26523	1.22
Dv6	1.25	1.29415	3.53
Dv7	1.25	1.32790	6.23

Table 17.4 What-if Results and Verification

Performance Measure	Current Value	Predicted Value	FEA Results	Accuracy
Volume	361.941 $in.^3$	376.345 $in.^3$	376.339 $in.^3$	100.0
Displacement	0.108173 in.	0.095420 in.	0.096425 in.	101.0

design change to reduce the maximum deformation at node 266. Table 17.4 shows the first-order prediction of displacement and volume performance values using the design sensitivity coefficients and design perturbation. It is shown that the displacement performance is reduced from 0.1082 to 0.0954 in., following such a design change. However, volume increases from 361.94 to 376.35 $in.^3$. A finite element analysis is carried out at the perturbed design to verify if the predictions are accurate. The finite element analysis results given in Table 17.4 show that the predicted performance values are very close to the results from finite element analysis because the sensitivity coefficients are accurate and the design perturbation is within a small range.

17.10.1.6 Trade-off Determination

From the design sensitivity displays and what-if study, a conflict is found in the design in reducing structural volume and maximum deformation. To find the best design direction, a trade-off study is carried out. To support the trade-off study, the volume performance measure is selected as the objective function, and the displacement performance measure is defined as constraint function, with an upper bound of 0.1 in. Notice that, in the current design, the displacement performance measure is 0.1082 in., which is greater than the bound. Therefore, the current design is infeasible. The side constraints are defined for all the design variables with bounds 0.1 and 10.0 in.

With an infeasible design, the second option, constraint correction, is selected for trade-off study. A QP subproblem is formed and solved to determine a design direction. Table 17.5 shows the design direction obtained from solving the QP subproblem.

A what-if study is carried out again following the design direction suggested by the trade-off determination, using a step size 0.1 in. The results of the what-if study are listed in Table 17.6, which show the approximation of objective and constraint function values using the design sensitivity coefficients and design perturbation. In this case, constraint violation is completely corrected with a

Table 17.5 Design Direction for Trade-off Determination

Design Variables	Current Value (in.)	Direction (in.)	Perturbation (in.)
Dv1	0.58	0.2305	0.0231
Dv2	0.58	0.07805	0.0078
Dv3	0.58	0.05536	0.0055
Dv4	0.58	0.09807	0.0098
Dv5	1.25	0.09759	0.0097
Dv6	1.25	0.02830	0.0028
Dv7	1.25	0.4992	0.0499

Table 17.6 What-if Study Results and Verification (from a Trade-Off Study)

Objective and Constraint	Current Value	Predicted Value	FEA Results	Accuracy
Objective	361.941 $in.^3$	371.172 $in.^3$	371.169 $in.^3$	100.0
Constraint	0.1082 in.	0.1000 in.	0.1004 in.	100.4

small increment in the objective function (volume). A finite element analysis is carried out at the perturbed design to verify the approximations are accurate, as given in Table 17.6.

17.10.1.7 Design Optimization

To perform design optimization, the same objective, constraint, and side constraints defined in the trade-off determination are used. DOT, ANSYS, and sensitivity computation and model update programs similar to those discussed in Section 17.8.1 are integrated to perform design optimization. After four iterations, a local minimum is achieved. The optimization histories for objective, constraint, and design variables are shown in Figures 17.28(a)–(c), respectively.

From Figure 17.28(a), the objective function starts around 362 $in.^3$ and jumps to 382 $in.^3$ immediately to correct constraint violation. Then, the objective function is reduced further until a minimum point, 354 $in.^3$ is reached. Also, the constraint function history graph shows that 80% violation is found at the initial design, and the violation is reduced significantly to 65% below the bound at the first iteration. Then, the constraint function is stabilized and stays feasible for the rest of the iterations. At optimum, the constraint is 4% below the bound, the maximum displacement becomes 0.09950 in., and the design is feasible. The most interesting observation is that, from Figure 17.28(c), all design variables are decreasing in the design iterations, except for dv1 and dv7. Design variable dv1 increases from 0.58 to 0.65 in. at the optimum. However, the most significant design change is dv7 from 1.25 to 1.44 in., which contributes largely to the reduction in the deformation, as listed in Table 17.7. Decrement of the rest of the design variables contributes to the volume reduction.

17.10.1.8 Postoptimum Study

At optimum, the designer can still acquire significant information to assist in the design or manufacturing process. The design sensitivity plot for a displacement performance measure at optimum is shown in Figure 17.29. The sensitivity plot shows that thickness at the outer edge has a significant effect on the maximum displacement at the contact area. In other areas, sensitivity coefficients are relatively small. This plot suggests that in the manufacturing process, restrictive tolerance needs to be applied to the thickness at the outer edge because small errors made in the outer rim will impact the displacement.

17.10.2 SHAPE OPTIMIZATION OF THE ENGINE CONNECTING ROD

This case study involves shape optimization of an engine connecting rod, in which the geometric shape of the rod is parameterized to achieve an optimal design considering essential structural performance measures, such as stresses. In this case, geometric and finite element models are created in a *p*-version

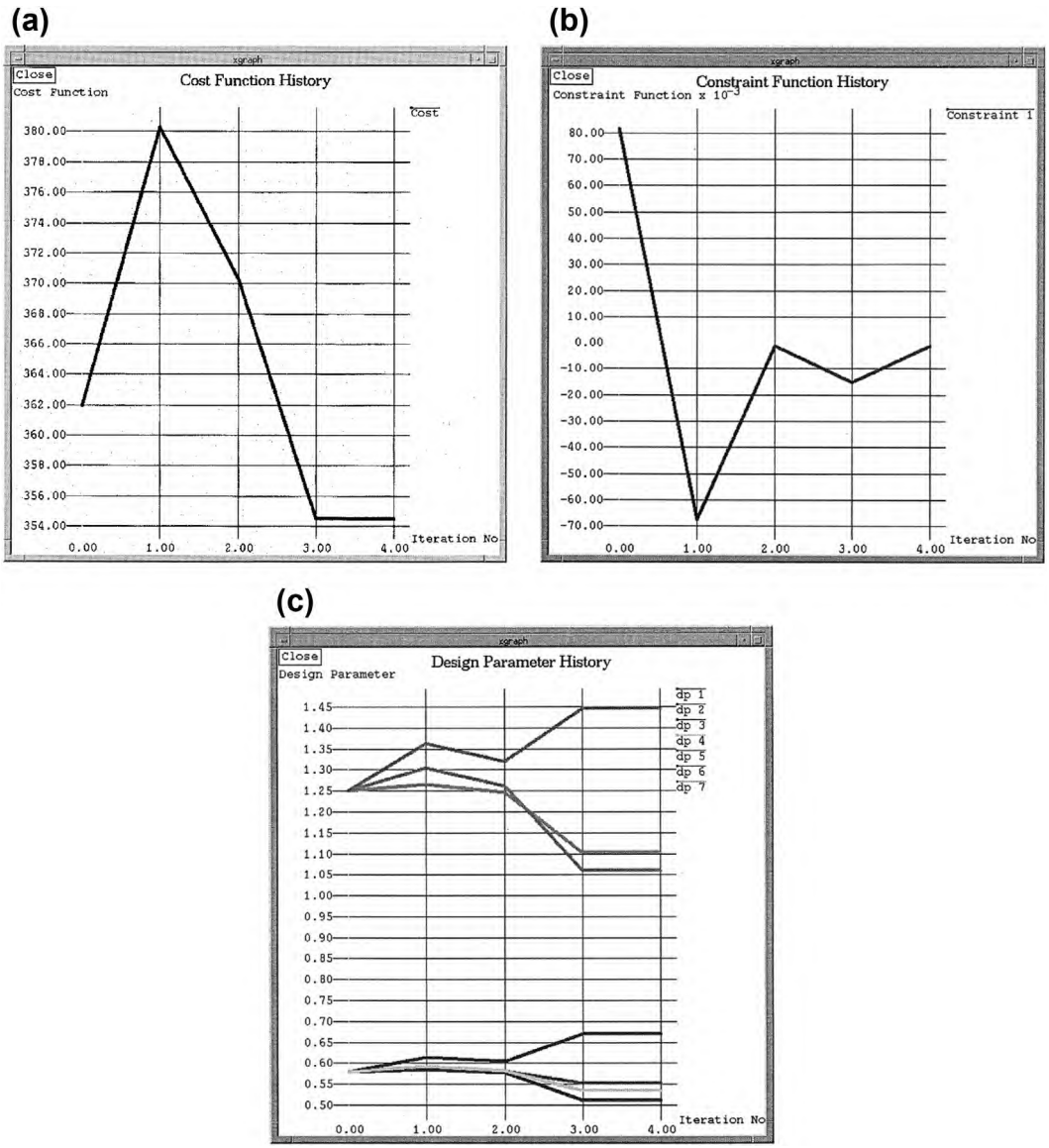


FIGURE 17.28

Design optimization history: (a) objective function, (b) constraint function, and (c) design variables.

FEA code, called STRESSCHECK (www.esrd.com). FEA is performed also using STRESSCHECK. Shape design sensitivity computation is carried out using the continuum-based method to be discussed in Chapter 18, and optimization is carried out using DOT. STRESSCHECK and DOT are integrated with the sensitivity analysis code to support a batch mode optimization for this example.

Table 17.7 Design Variable Values at Optimum

Design Variables	Initial Design (in.)	Optimum Design (in.)
dv1	0.58	0.650
dv2	0.58	0.556
dv3	0.58	0.512
dv4	0.58	0.540
dv5	1.25	1.113
dv6	1.25	1.053
dv7	1.25	1.442

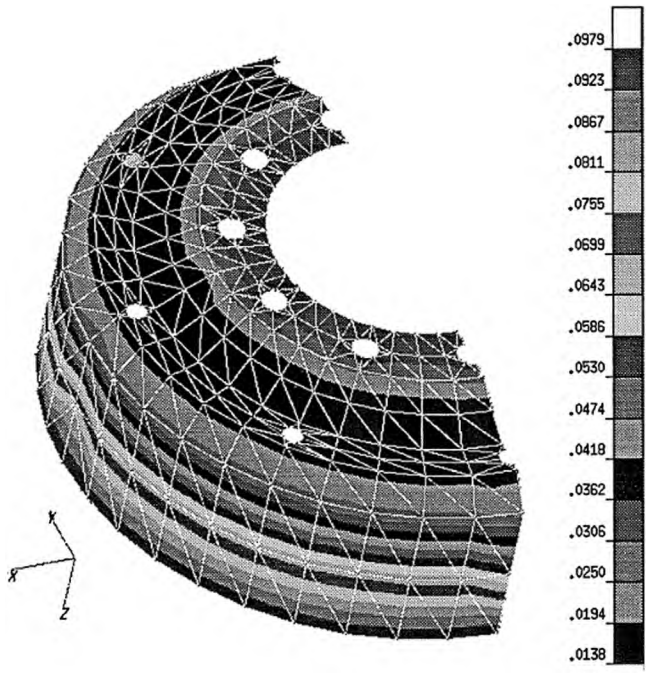
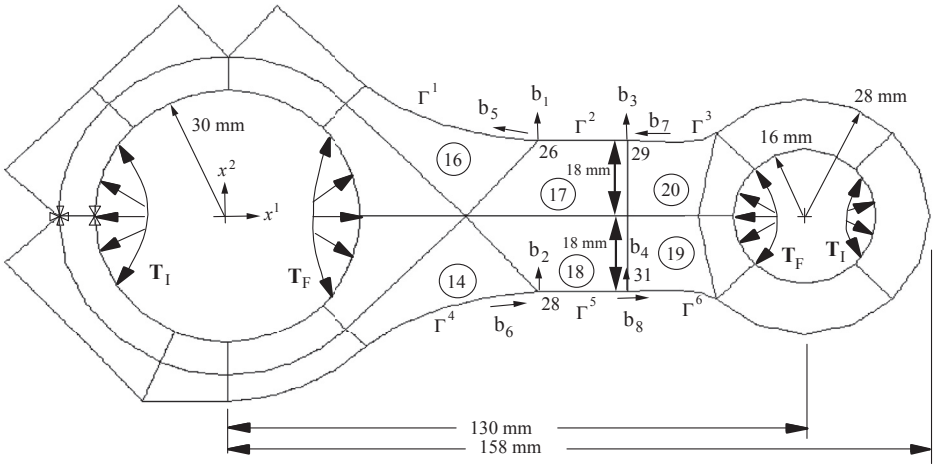


FIGURE 17.29

Design sensitivity of displacement performance measure at the optimum displayed in fringe plot in PATRAN.

17.10.2.1 Geometric and Finite Element Models

The connecting rod is modeled as a plane stress problem using 25 quadrilateral p-elements. The shape of the connecting rod is to be determined to minimize its weight subject to a set of stress constraints. The geometric model, finite element mesh, physical dimensions, and design variables are shown in Figure 17.30. The material properties are modulus of elasticity $E = 2.07 \times 10^5$ MPa and Poisson's ratio $v = 0.298$.



$E = 2.07 \times 10^5$ MPa, $\nu = 0.298$, $\rho = 7.81 \times 10^{-6}$ kg/mm^3, Plane Stress Problem

: movement x^2-direction is fixed, : movements x^1-and x^2-direction are fixed

Γ^1, Γ^2, Γ^3, Γ^4, Γ^5, Γ^6 : design boundaries

(14) (16) (17) (18) (19) (20) : finite elements with design boundaries

b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8 : design parameters

FIGURE 17.30

The engine connecting rod model (Hwang et al., 1997).

In this problem, two loadings are considered: the firing load T_F, which occurs during the combustion cycle, and the inertia load T_I, which occurs during the suction cycle of the exhaust stroke. These loads are defined as follows (Hwang et al., 1997):

$$T_F = \begin{cases} 357.589\theta^2 - 0.0263131\theta - 175.390, & \text{at left inner circle}, \quad -40^\circ \le \theta \le 40^\circ \\ 518.622\theta^2 - 3258.60\theta + 4812.67, & \text{at right inner circle}, \quad 140^\circ \le \theta \le 220^\circ \end{cases} \tag{17.98}$$

$$T_I = \begin{cases} 21.7327\theta^4 - 282.180\theta^3 - 1335.71\theta^2 - 2723.33\theta + 1998.7, \\ \quad \text{at left inner circle}, \quad 105^\circ \le \theta \le 225^\circ \\ 49.6133\theta^4 1.22975\theta^3 - 76.8156\theta^2 - 0.823978\theta 12.4547, \\ \quad \text{at right inner circle}, \quad -75^\circ \le \theta \le 75^\circ \end{cases} \tag{17.99}$$

where θ is the angle measured counterclockwise from the positive x^1-axis.

17.10.2.2 Design Parameterization and Problem Definition

A design boundary that consists of boundary segments Γ^1 to Γ^6 is parameterized using Hermit cubic curves. Eight design variables are shown in Figure 17.30 and listed in Table 17.8. For the firing load,

Table 17.8 Design Variables of Engine Connecting Rod

Design Variable	Definition
b_1	Position of node 26 in x^2-direction
b_2	Position of node 28 in x^2-direction
b_3	Position of node 29 in x^2-direction
b_4	Position of node 31 in x^2-direction
b_5	Slope at node 26
b_6	Slope at node 28
b_7	Slope at node 29
b_8	Slope at node 31

the upper bound of the allowable principal stress is $\sigma_{UF} = 37$ MPa, and the lower bound is $\sigma_{UF} = -279$ MPa. For the inertia load, the upper bound of the allowable principal stress is $\sigma_{UI} = 136$ MPa, and the lower bound is $\sigma_{UI} = -80$ MPa. There are 488 stress constraints imposed along boundaries Γ^1 to Γ^6 and at the interior of elements 14, 16, 17, 18, 19, and 20, as shown in Figure 17.31.

17.10.2.3 Design Optimization

At the initial design, the total weight of the rod is 0.5932 N and no stress constraints are violated. For design optimization, the modified feasible direction method of DOT is used. After five design iterations, an optimum design is obtained. The total weight at the optimum design is 0.5298 N. The objective function history is shown in Figure 17.32. The total weight has been reduced by 10.7%. Because the firing load is larger than the inertia load, a number of stress constraints are active under the firing load at the optimum design. Figures 17.33 and 17.34 are the stress contours at the initial and optimum designs, respectively. As shown in Figure 17.34, all active stress constraints are due to the firing load.

Figure 17.35 shows the design and finite element models at the initial and optimum designs. A reduced optimum weight is obtained by distributing the high stress points along the design boundary Γ^1 to Γ^6 and making stress constraints active. At the optimum design, the neck region become thinner while stress constraints are not violated.

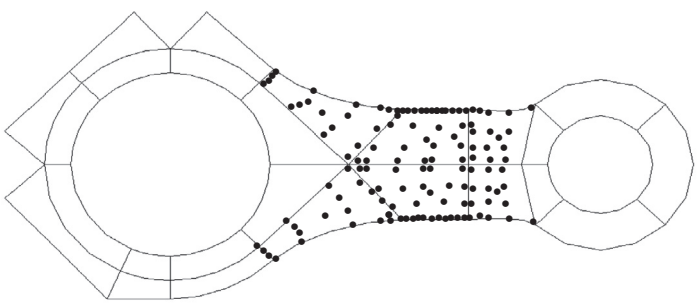


FIGURE 17.31

Locations of stress constraint points of the connecting rod.

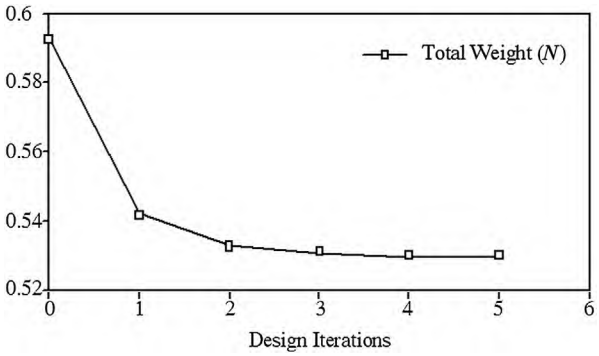


FIGURE 17.32

Objective function history of connecting rod.

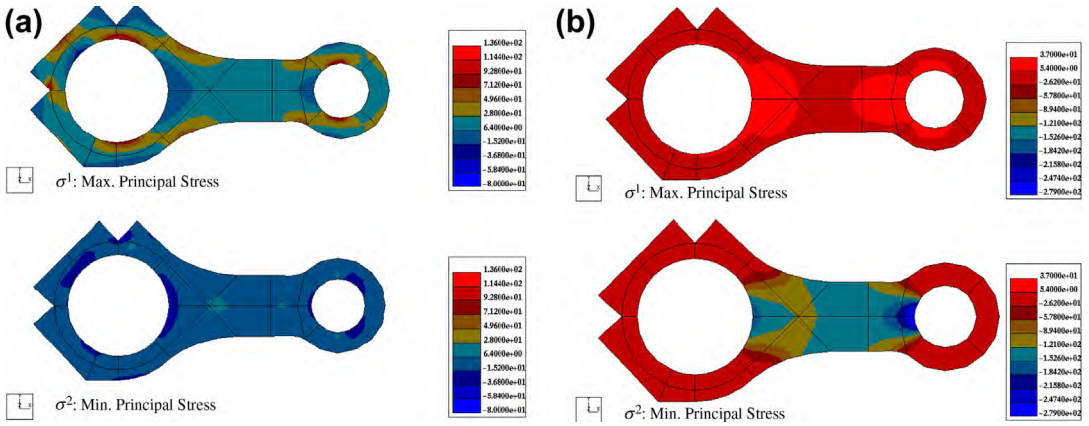


FIGURE 17.33

Stress fringe plots at initial design: (a) due to inertia load and (b) due to firing load.

17.11 TUTORIAL EXAMPLE: SIMPLE CANTILEVER BEAM

We use the cantilever beam example shown in Figure 17.4 to illustrate detailed steps for carrying out design optimization using SolidWorks Simulation and Pro/MECHANICA Structure. The beam is made of aluminum (2014 Alloy) with modulus of elasticity $E = 1.06 \times 10^7$ psi and Poisson's ratio $\nu = 0.33$. At the current design, the beam length is $\ell = 10$ in., and the width and height of the cross-section are both 1 in. The load is $P = 1000$ lbf acting at the tip of the beam. Our goal is to optimize the beam design for a minimum volume subject to displacement and stress constraints by varying its length as well as the width and height of the cross-section.

The design problem of the cantilever beam example is formulated mathematically as

$$\text{Minimize}: \quad V(w, h, \ell) = wh\ell \tag{17.100a}$$

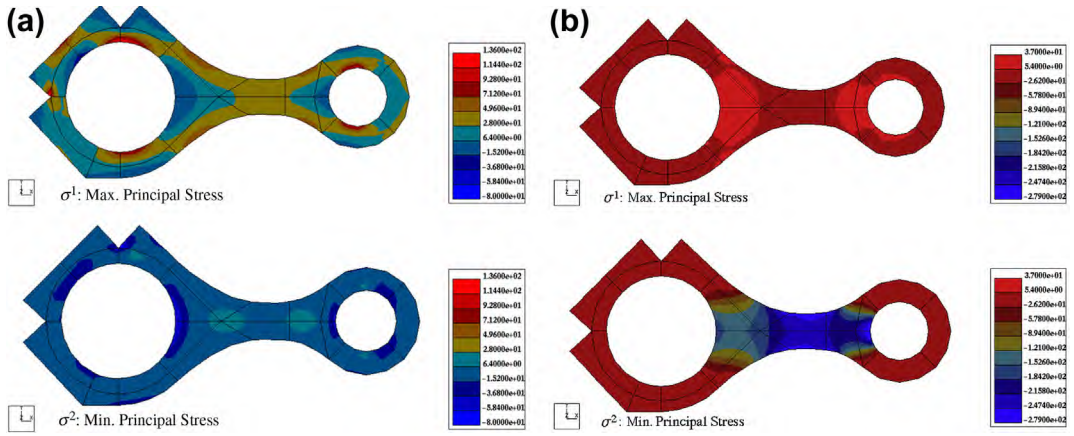


FIGURE 17.34

Stress fringe plots at optimal design: (a) due to inertia load and (b) due to firing load.

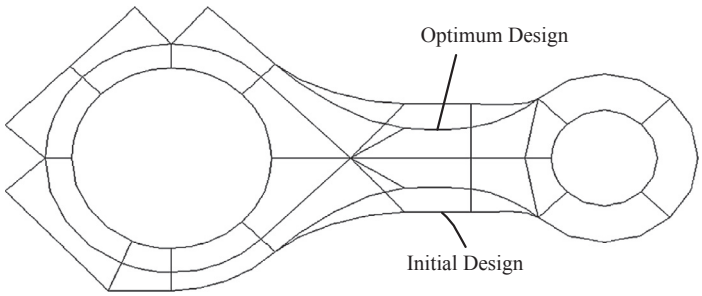


FIGURE 17.35

Geometry model of the connecting rod at initial and optimum designs.

$$\text{Subject to:} \quad g_1(w,h,\ell) = \sigma_{\text{mp}}(w,h,\ell) - 65 \text{ ksi} \leq 0 \tag{17.100b}$$

$$g_2(w,h,\ell) = \delta_y(w,h,\ell) - 0.1 \text{ in.} \leq 0 \tag{17.100c}$$

$$0.5 \text{ in.} \leq w \leq 1.5 \text{ in.}, 0.5 \text{ in.} \leq h \leq 1.5 \text{ in.}, 5 \text{ in.} \leq \ell \leq 15 \text{ in.} \tag{17.100d}$$

in which σ_{mp} is the maximum principal stress, and δ_y is the maximum displacement in the y-direction (vertical).

This design problem will be implemented and solved using both Simulation and Mechanica in Projects S5 and P5, respectively. We briefly discuss the example in this section. Detailed step-by-step operations in using the software tools can be referred to the respective projects.

17.11.1 USING SOLIDWORKS SIMULATION

SolidWorks Simulation employs a generative method for support of design optimization. Design variables vary between their respective lower and upper bounds. These design variables are combined to create individual design scenarios. Finite element analyses are carried out for all scenarios generated. Among the scenarios evaluated, feasible designs are collected; and within the feasible designs, the best design that yields the lowest value in the objective function is identified as the solution to the design problem.

In the beam example, we chose a 0.5 in. interval for varying the width and height design variables, and a 5 in. interval for the length design variable. As a result, each design variable is varied three times. For example, the width design variable is changed from its lower bound of 0.5 in. to 1.0 in., and then to its upper bound 1.5 in. Similarly, three changes take place for the height and length design variables, respectively. These changes in design variables are combined to create 27 ($3 \times 3 \times 3$) scenarios to search for an optimal solution.

A static design study is created for the beam with boundary and load conditions shown in Figure 17.36. Also shown in Figure 17.36 are the finite element mesh (default median mesh) and the fringe plot of the maximum principal stress (or first principal stress).

A total of 29 static analyses (27 plus initial and current designs) are carried out for the study. In the graphics window, the designs of all scenarios will appear in the beam with changing dimensions, as shown in Figure 17.37.

At the end, an optimal solution is found for Scenario 19 (see Figure 17.38), in which width, height, and length become 0.5, 5, and 1.5 in., respectively. Displacement is 0.0296 in. (<0.1 in.), stress is 42,076 psi (<65 ksi), and the total mass is 0.379 lb (reduced from 1.01 lb from initial design).

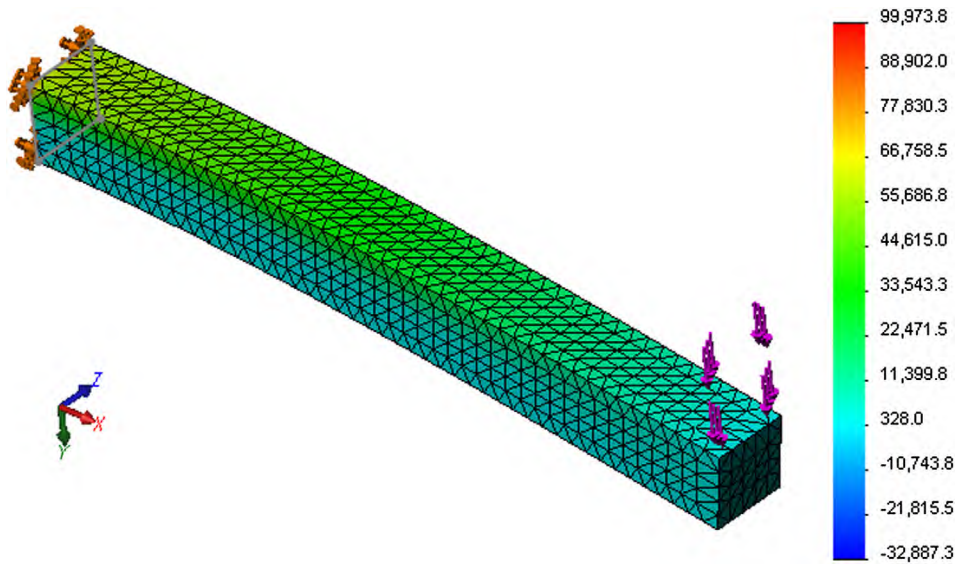


FIGURE 17.36

Finite element model of the cantilever beam in SolidWorks Simulation.

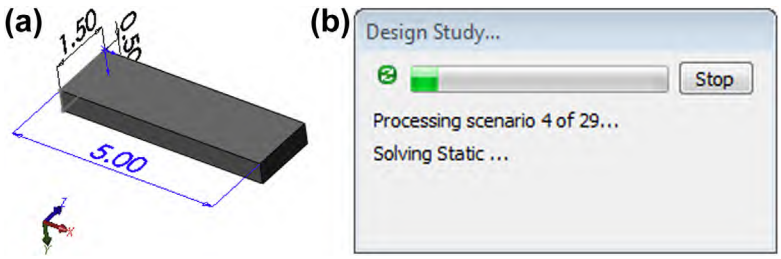


FIGURE 17.37

Design optimization underway: (a) beam with varying dimensions and (b) status dialog box showing progress.

29 of 29 scenarios ran successfully. Design Study Quality: High

		Current	Initial	Optimal (19)	Scenario 16	Scenario 17	Scenario 18	Scenario 19	Scenario 20	Scenario 21
Length		10in	10in	5in	5in	10in	15in	5in	10in	15in
Width		1in	1in	0.5in	1.5in	1.5in	1.5in	0.5in	0.5in	0.5in
Height		1in	1in	1.5in	1in	1in	1in	1.5in	1.5in	1.5in
Stress1	< 65000 psi	99974 psi	99974 psi	42076 psi	28387 psi	67068 psi	83643 psi	42076 psi	89354 psi	1.1789e+005 psi
Displacement1	< 0.1in	0.37721in	0.37721in	0.0296in	0.03171in	0.25065in	0.84573in	0.0296in	0.22621in	0.75782in
Mass1	Minimize	1.01156 lb	1.01156 lb	0.379336 lb	0.758673 lb	1.51735 lb	2.27602 lb	0.379336 lb	0.758673 lb	1.13801 lb

FIGURE 17.38

Optimal solution reported by Simulation.

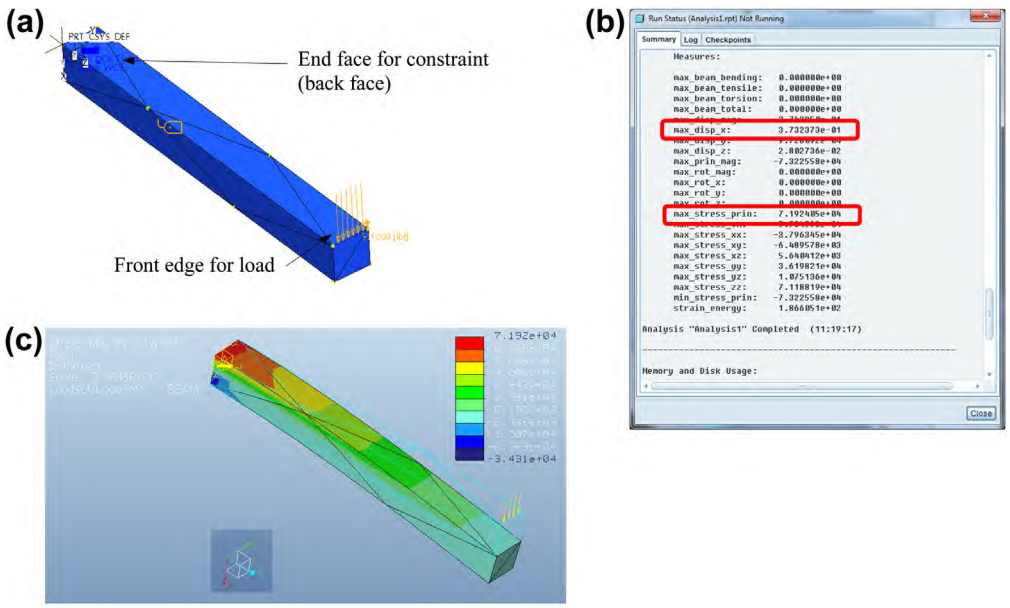


FIGURE 17.39

Finite element model of the cantilever beam in Mechanica: (a) boundary and load condition with mesh, (b) status dialog box with measures, and (c) maximum bending stress fringe plot.

17.11.2 USING PRO/MECHANICA STRUCTURE

Unlike Simulation, Mechanica employs a gradient-based solution technique for solving design problems. Accessing the optimization capability in Mechanica is similar to that of defining and solving an FEA, which is straightforward.

A static design study is created for the beam with boundary and load conditions shown in Figure 17.39(a). Figure 17.39(a) also shows the finite element mesh (13 tetrahedron solid elements). The maximum bending stress fringe plot is shown in Figure 17.39(c), which shows maximum tensile and compressive stresses, respectively, at the top and bottom fibers of the beam close to the root end. FEA results are also given in the status dialog box shown in Figure 17.39(b), in which maximum displacement and principal stress are 0.373 in. and 71.92 ksi, respectively.

An optimal solution is obtained in seven design iterations. At the optimum, the three design variables are length $\ell = 5$ in., width $w = 0.5$ in., and height $h = 1.08$ in. Constraint functions are stress $\sigma_{mp} = 64.99$ ksi, and displacement $\delta_x = 0.0766$ in.; both are feasible. The total mass is 0.0007056 lbf s^2/in. (or 0.272 lbm), reduced from 0.002669 lbf s^2/in. (or 1.03 lbm) from the initial design. The optimization history graphs for objective and constraint functions are shown in Figures 17.40(a)–(c), respectively.

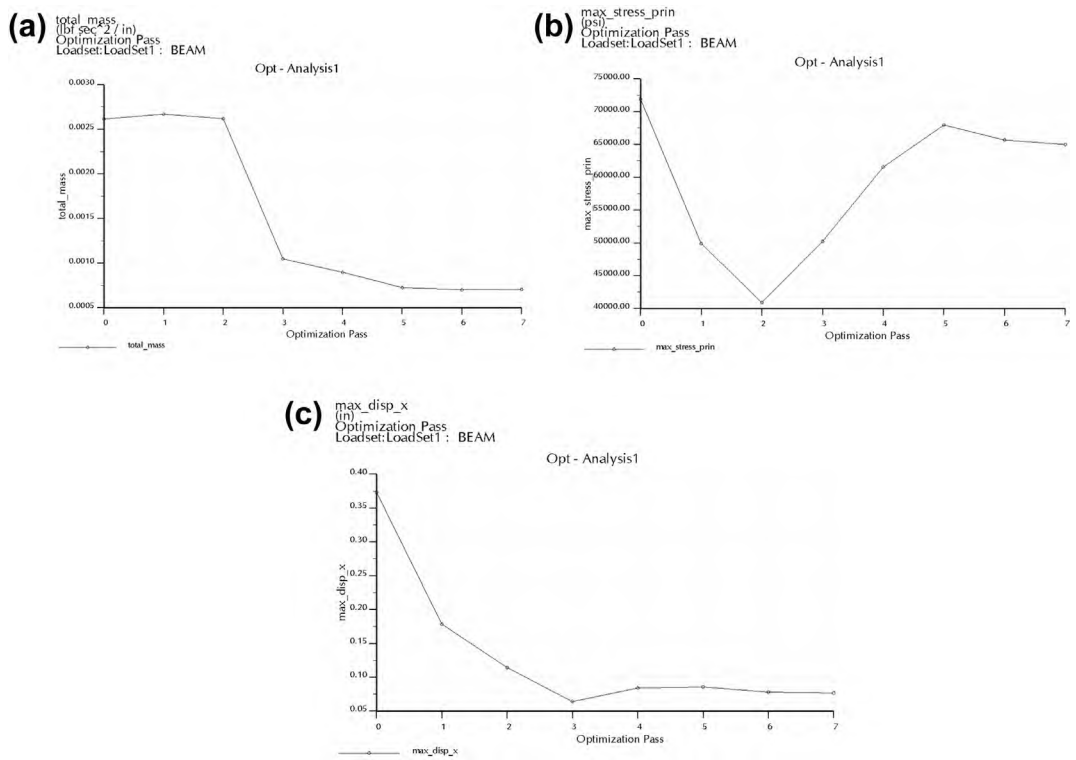


FIGURE 17.40

The optimization history graphs: (a) objective function, (b) stress constraint function, and (c) displacement constraint function.

17.12 SUMMARY

In this chapter, we discussed a broad range of topics in design optimization. We started with a simple beer can design problem, then moved into basic optimization concepts—specifically, the optimality conditions. We introduced three major solution approaches: optimality criteria, the graphical method, and the search method, for both constrained and unconstrained problems of linear and nonlinear functions. We pointed out that the search methods using numerical algorithms, including gradient-based and non-gradient approaches, are most general. We also mentioned that the gradient-based approach requires far fewer function evaluations; therefore, it is better suited to large-scale problems. We discussed several classical and widely accepted algorithms of the gradient-based search methods for both constrained and unconstrained problems. We also provided simple examples for explaining the concept and illustrating details of these algorithms. We included two representative algorithms for the non-gradient approach: genetic algorithm and simulated annealing. With a basic understanding of the concept and solution techniques of design optimization, we also discussed practical aspects in solving practical design problems using commercial CAD and CAE codes. We reviewed commercial CAD, CAE, and optimization tools, which offer plausible capabilities for solving general design optimization problems. We also presented two case studies that examined more in-depth in the kind of practical problems that the optimization techniques are able to support and how they are solved using commercial tools and in-house software modules.

We hope this chapter provided readers with adequate depth and breadth on this important and widely employed topic in engineering design. As we pointed out numerous times, the gradient-based solution techniques are most suitable to solving general engineering problems, especially those requiring substantial computation time. The key ingredient of the gradient-based techniques is the gradient calculation, also called design sensitivity analysis.

With a basic understanding of the subject of design optimization, we narrow our focus into structural design in Chapter 18, in which we discuss structural design problems, including sizing, material, shape, and topology. We introduce design sensitivity analysis methods and integration of sensitivity analysis with FEA and optimization algorithms for solving structural optimization problems. Looking ahead to Chapter 19, we discuss MOO encountered in practices, especially when we solve design problems that involve multiple engineering disciplines. Important topics to be discussed in Chapters 18 and 19 should help us become more capable and competent in solving engineering design problems.

QUESTIONS AND EXERCISES

17.1. A function of two variables is defined as

$$f(x_1, x_2) = \left(\cos^2 x_1 + \cos^2 x_2\right)^2$$

Sketch by hand or use MATLAB to graph the function, and calculate the gradient vectors of the function at $(x_1, x_2) = (0, 0)$, $(x_1, x_2) = (1, -1)$, and $(x_1, x_2) = (-1, 1)$. Sketch the gradient vectors on the x_1–x_2 plane together with their respective iso-lines.

17.2. Design a rectangular box with a square cross-sectional area of width x and height h. The goal of the design problem is to maximize the volume of the box subject to the sum of the length

of the 12 edges to a fixed number, say, $L = 192$. Formulate the design problem as a constrained optimization problem. Convert the constrained problem to an unconstrained problem and solve the unconstrained problem by finding stationary point(s) and checking the second derivatives of the objective function.

17.3. Continue with Problem 17.2. First, convert the constrained problem into an unconstrained problem using Lagrange multiplier and solve the optimization problem by using the KKT conditions.

17.4. A function of three variables is defined as

$$f(x_1, x_2, x_2) = 3x_1^2 + 2x_1x_2 + 8x_2^2 + x_3^2 - 2x_1 + 5x_2 - 9$$

Calculate the gradient vector of the function, and determine if a stationary point exists. If it does, calculate a Hessian matrix of the function f, and determine if the stationary point found gives a minimum value of the function f.

17.5. Solve the following LP problems using the graphical method.
Problem (a)

$$\begin{aligned} \text{Minimize:} \quad & f(\mathbf{x}) = 4x_1 - 5x_2 \\ \text{Subject to:} \quad & g_1(\mathbf{x}) = -4 - x_1 + x_2 \le 0 \\ & g_2(\mathbf{x}) = -7 + x_1 + x_2 \le 0 \end{aligned}$$

Problem (b)

$$\begin{aligned} \text{Minimize:} \quad & f(\mathbf{x}) = 2x_1 - x_2 \\ \text{Subject to:} \quad & g_1(\mathbf{x}) = -12 + 4x_1 + 3x_2 \le 0 \\ & g_2(\mathbf{x}) = -4 + 2x_1 + x_2 \le 0 \\ & g_3(\mathbf{x}) = -4 + x_1 + 2x_2 \le 0 \\ & 0 \le x_1, 0 \le x_2 \end{aligned}$$

17.6. Solve the following NLP optimization problems using the graphical method.
Problem (a)

$$\begin{aligned} \text{Minimize:} \quad & f(\mathbf{x}) = x_1^2 - 3x_1x_2 + x_2^2 \\ \text{Subject to:} \quad & g_1(\mathbf{x}) = x_1^2 + x_2^2 - 6 \le 0 \\ & 0 \le x_1, \text{ and } 0 \le x_2 \end{aligned}$$

Problem (b)

$$\begin{aligned} \text{Minimize:} \quad & f(\mathbf{x}) = 2x_1^3 - 8x_1x_2 + 15x_2^2 - 4x_1 \\ \text{Subject to:} \quad & h_1(\mathbf{x}) = x_1^2 + x_1x_2 + 1 = 0 \\ & g_1(\mathbf{x}) = 4x_1 + x_2^2 - 4 \le 0 \end{aligned}$$

17.7. Use the golden search method to find the minimum point of the following functions in the interval of $1 \leq x \leq 10$. We assume the convergent tolerance is $\varepsilon_1 = 0.0001$.

$$\text{Function (a)}: f(x) = x^2 + 500/x^3$$

$$\text{Function (b)}: f(x) = x^2 - e^{x^2+1}$$

17.8. Use the gradient-based method to find the minimum point of the same objective functions (a) and (b) as Problem 7. We assume the initial design is given at $x^0 = 1$ and convergent tolerance is $\varepsilon_1 = 0.0001$. Use a proper method to determine the step size.

17.9. Use the steepest descent method to find the minimum point of the objective function

$$f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 3x_3^2 + 2x_1x_2 + 7x_2x_3$$

We assume the initial design is given at $\mathbf{x}^0 = (2, 4, 5)$, the initial step size $\alpha = 1$, and convergent tolerance is $\varepsilon_1 = 0.00001$

17.10. Use the conjugate gradient method to find the minimum point of the same function of Problem 17.9.

17.11. Use the BFGS method to find the minimum point of the same objective function of Problem 17.9.

17.12. Use the BFGS method combined with the Newton's method for a line search to find the minimum point of the same objective function of Problem 17.9.

17.13. Continue with Problem 17.12, except using the secant method for a line search.

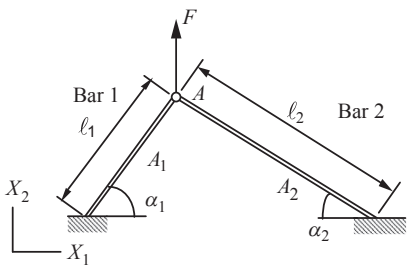
17.14. Solve Problems 17.6(a) and 17.6(b) using SLP. Determine adequate move limits.

17.15. Solve Problem 17.14, except using SQP.

17.16. Solve Problem 17.14, except using the feasible direction method.

17.17. Solve Problem 17.14, except using the penalty method.

17.18. A simple two-bar truss structure shown below supports a vertical force F without structural failure. The cross-sectional areas of the bars are A_1 and A_2, and the lengths of the bars are ℓ_1 and ℓ_2. Cross-sectional areas of the bars are allowed to change.



To simplify the physical model, the structure is assumed to be homogeneous with material properties: modulus of elasticity E and mass density ρ. Also, the bars are assumed to be straight before and after deformation; that is, no bending effect is considered in this problem. Because the problem is simple and can be solved analytically, there is no need to use finite element method.

A design problem is formulated as:

$$\text{Minimize:} \quad M = \rho_1 A_1 \ell_1 + \rho_2 A_2 \ell_2 \tag{a}$$

$$\text{Subject to:} \quad \sigma_1 \leq \overline{\sigma} \tag{b}$$

$$\sigma_2 \leq \overline{\sigma} \tag{c}$$

$$0 < A_1 \tag{d}$$

$$0 < A_2 \tag{e}$$

where M is mass of the structure; σ_1 and σ_2 are the stresses in bar 1 and bar 2, respectively, which must be less than the stress failure limit $\overline{\sigma}$; and the areas must be positive.

a. Sketch a free body diagram of force acting at point A, as shown in the figure above, formulate equilibrium equations, and solve for the stresses σ_1 and σ_2.

b. Calculate the derivatives of the objective and stress constraint functions with respect to design variables A_1 and A_2, respectively.

c. Sketch the feasible region and iso-lines of the objective function on the A_1–A_2 plane and find the optimal solution to the optimization problem using the graphical method. The numerical numbers of the properties are given as follows: modulus of elasticity $E = 2.1 \times 10^5$ MPa, Poisson's ratio $v = 0.29$, mass density $\rho_1 = \rho_2 = 7.8 \times 10^{-6}$ kg/mm^3, $\ell_1 = 300$ mm, $\alpha_1 = 45°$, $\alpha_2 = 30°$, $A_1 = A_2 = 10$ mm^2, $\overline{\sigma} = 45$ MPa, and $F = 1000$ N

REFERENCES

Arora, J.S., 2012. Introduction to Optimum Design. Academic Press.

Chang, K.H., Joo, S.H., 2006. Design parameterization and tool integration for CAD-based mechanism optimization. Advanced Engineering Software 37, 779–796.

Chang, K.H., Choi, K.K., Perng, J.H., 1992. Design sensitivity analysis and optimization tool for sizing design applications. In: Fourth AIAA/AIR Force/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, Paper No. 92-4798, pp. 867–877. Cleveland, Ohio.

Chang, K.H., Choi, K.K., 1993. Shape design sensitivity analysis and optimization for spatially rotating objects. Journal of Structural Optics 6 (4), 216–226.

Clark, F., Lorenzoni, A.B., 1996. Applied Cost Engineering, third ed. CRC.

Fletcher, R., Reeves, R.M., 1964. Function minimization by conjugate gradients. Comput. J 7, 149–160.

Hardee, E., Chang, K.H., Tu, J., Choi, K.K., Grindeanu, I., Yu, X., 1999. CAD-based shape design sensitivity analysis and optimization. Advanced Engineering Software 30 (3), 153–175.

Haug, E.J., Smith, R.C., 1990. DADS-Dynamic Analysis and Design System. Multibody Systems Handbook, pp. 161–179.

Hwang, H.Y., Choi, K.K., Chang, K.H., 1997. Shape design sensitivity analysis and optimization using p-version finite element analysis. Mechanism of Structural Machine 25 (1), 103–137.

Kouvelis, P., Yu, G., 1997. Robust Discrete Optimization and Its Applications. Kluwer Academic Publishers, P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

Ostwald, P.F., 1991. Engineering Cost Estimating, third ed. Prentice Hall.

Ostwald, P.F., McLaren, T.S., 2004. Cost Analysis and Estimating for Engineering and Management. Pearson/Prentice Hall.

PolyFEM Development Group, 1994. PolyFEM-Pro/ENGINEER User Guide and Reference. Release 2.0. IBM Almaden Research Center, San Jose, CA.

Rao, S.S., 2009. Engineering Optimization: Theory and Practice, fourth ed. Wiley.

Syslo, M.M., Deo, N., Kowalik, J.S., 1983. Discrete Optimization Algorithms: With Pascal Programs. Dover Publications, Inc., Mineola, NY.

Vanderplaats, G.N., 2005. Numerical Optimization Techniques for Engineering Design, fourth ed. Vanderplaats Research & Development, Inc.

Vanderplaats, G.N., 2007. Multidiscipline Design Optimization. Vanderplaats Research & Development, Inc., Colorado Springs, CO.

Vincent, T.L., 1983. Game theory as a design tool. Journal of Mechanical Transmission 105, 165–170.

CHAPTER

STRUCTURAL DESIGN SENSITIVITY ANALYSIS

18

CHAPTER OUTLINE

18.1 Introduction 1003
18.2 Simple Bar Example 1005
18.2.1 Differential Equation 1006
18.2.2 Energy Equation 1007
18.2.3 Finite Element Formulation 1009
18.3 Sensitivity Analysis Methods 1012
18.3.1 Analytical Derivative Method 1014
18.3.2 Overall Finite Difference 1015
18.3.3 Discrete Approach 1017
18.3.3.1 Direct Differentiation Method 1019
18.3.3.2 Adjoint Variable Method 1022
18.3.4 Continuum Approach 1024
18.3.4.1 Direction Differentiation Method 1025
18.3.4.2 Adjoint Variable Method 1027
18.4 Sizing and Material Designs 1029
18.4.1 Principle of Virtual Work 1030
18.4.1.1 Differential Equation 1030
18.4.1.2 Energy Formulation 1032
18.4.2 Variations 1033
18.4.2.1 Definition 1033
18.4.2.2 Variations of the Energy Bilinear and Load Linear Forms 1036
18.4.3 Static Problems 1037
18.4.3.1 Energy Formulation 1038
18.4.3.2 Finite Element Discretization 1040
18.4.3.3 Direct Differentiation Method of the Continuum-Discrete Approach 1043
18.4.3.4 Sensitivity of the Bending Stress 1044
18.4.3.5 Adjoint Variable Method of the Continuum-Discrete Approach 1046
18.4.3.6 Adjoint Variable Method of the Continuum-Analytical Approach 1050
18.4.4 Numerical Implementation 1052
18.5 Shape Sensitivity Analysis* 1053
18.5.1 Domain Shape Sensitivity Analysis 1053
18.5.2 A Simple Cantilever Beam Example 1054

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00018-1
Copyright © 2015 Elsevier Inc. All rights reserved.

18.5.3 Shape Design Parameterization 1058
18.5.3.1 2-D Planar Structures 1058
18.5.3.2 3-D Solid Structures—Freeform Surfaces 1061
18.5.3.3 3-D Solid Structures—CAD-Generated Surfaces 1063
18.5.4 Design Velocity Field Computation 1065
18.5.4.1 Design Velocity Field 1066
18.5.4.2 Boundary Velocity Computation 1068
18.5.4.3 Domain Velocity Computation 1075
18.5.5 Shape Sensitivity Analysis Using Finite Difference or Semi-Analytical Method 1080
18.5.6 Material Derivatives 1082
18.5.7 Shape Sensitivity Analysis Using the Continuum Approach 1087
18.5.7.1 Shape Sensitivity Analysis for a Cantilever Beam 1088
18.6 Topology Optimization 1090
18.6.1 Basic Concept and Problem Formulation 1091
18.6.2 Two-Dimensional Cantilever Beam Example 1093
18.7 Case Study 1094
18.7.1 The Tracked Vehicle Roadarm 1095
18.7.2 Topology Optimization 1095
18.7.3 Boundary Smoothing 1095
18.7.4 Shape Parameterization and Design Velocity Field Computation 1096
18.7.5 Shape Design Optimization 1097
18.8 Summary 1099
Questions and Exercises 1099
References 1102

In this chapter, we provide a broad but brief discussion on the topic of design sensitivity analysis (DSA), which calculates gradients (or derivatives) of structural performance measures with respect to design variables. Sensitivity analysis is essential for gradient-based optimization algorithms, in which accurate and efficient computation methods are desirable. In this chapter, we narrow our focus on structural problems, more specifically, linear elastic structures under static loads. We introduce basic concepts and solution methods in DSA with enough depth so that readers become familiar with this topic and are able to implement some of the techniques for support of engineering design. We include in this chapter basic topics that are relevant to structural problems, such as sizing, shape, and topology designs. We introduce a broad range of methods in sensitivity analysis, including the brute-force finite difference, popular semi-analytical methods, and mathematically rigorous and accurate continuum-based approaches. Some of the methods discussed can be easily extended to support engineering applications beyond structural problems; for example, the overall finite difference methods are general and can be implemented for problems such as mechanism design optimization (Chang and Joo, 2006). We also offer case studies to illustrate practical applications of the methods for structural problems, including a tracked vehicle roadarm that was designed first by using topology optimization and then later shape optimization.

Some of the discussions are quite mathematically involved. We assume readers are familiar with basic finite element analysis (FEA) theory and the relevant energy principles associated with it. We intend to use simple examples to illustrate basic concepts and solution techniques so as to avoid complex mathematical derivations. Readers who are proficient in FEA theory and desire to learn more about this topic, especially the continuum-based approaches, are referred to Choi and Kim (2006a,b).

The objectives of this chapter include (1) providing readers with the basic concepts and solution methods for structural sensitivity analysis and using simple examples to help readers grasp a basic understanding on the topic, (2) familiarizing readers with practical applications of the sensitivity analysis through a case study, and (3) providing a brief discussion on the implementation aspect of the continuum-discrete method to assist readers in future endeavors on this and other relevant topics.

Although only simple examples are employed for illustration, readers should be mindful that the concept and solution techniques introduced in this chapter are aimed at solving general and large-scale structural problems.

18.1 INTRODUCTION

As discussed in Chapter 17, numerical optimization techniques can be categorized as gradient-based and non-gradient algorithms. Gradient-based algorithms often lead to a local optimum. Non-gradient algorithms usually converge to a global optimum, but they require a substantial amount of function evaluations. For large-scale problems, as are often encountered in engineering design, non-gradient algorithms are less desirable. Gradient-based algorithms require gradient or sensitivity information, in addition to function evaluations, to determine adequate search directions for better designs during optimization iterations.

In optimization problems, the objective and constraint functions are often called performance measures. Sensitivity, sensitivity coefficient, and gradient are terms used interchangeably to define the rate of change in a performance measure with respect to the change in design variable. They support not only gradient-based optimization but reveal critical design information that could guide designers to achieve improved designs in just one or two design iterations—for example, through the interactive design steps discussed in Chapter 17.

As shown in Figure 18.1, a change in the height h or width w of a cantilever beam with the rectangular cross-section affects the maximum bending stress inside the beam. The bending stress can be formulated as

$$\sigma(w,h) = \frac{6P\ell}{wh^2} \tag{18.1}$$

where P is a point force applied at the tip of the beam. The gradient of the bending stress can be calculated by taking derivatives of the stress in Eq. 18.1 with respect to height and width of the beam cross-section, respectively, as

$$\frac{\partial \sigma}{\partial w} = -\frac{6P\ell}{w^2h^2} \tag{18.2a}$$

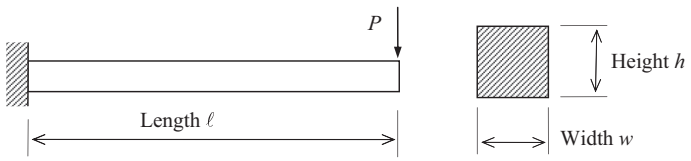


FIGURE 18.1

Cantilever beam with a rectangular cross-section.

and

$$\frac{\partial \sigma}{\partial h} = -\frac{12P\ell}{wh^3} \tag{18.2b}$$

From Eqs 18.2a and 18.2b, both gradients are negative, implying that increasing either height or width reduces the bending stress. Physically, it is obvious that increasing either height or width increases the moment of inertia I of the beam cross-section. In this case, $I = wh^3/12$, therefore reducing the bending stress. Between the two design variables, increasing the height dimension h is two times more effective than the width w in reducing the bending stress if $w = h$ (a square cross-section at the current design). Equations, such as Eqs 18.2a and 18.2b, provide desired information that supports the designer to achieve improved design effectively.

The formulation of DSA can vary significantly depending on the type of design variables being considered. In general, a structure consists of bars (or trusses), beams, membranes, shells, and/or elastic solid structural components. Depending on the constituents of the structure being designed, there are in general five different types of design variables—material, sizing, configuration, shape, and topology, as illustrated in Figure 18.2. For example, for a bar or beam structure shown in Figure 18.2(a), material design variables can be mass density ρ or modulus of elasticity E, while sizing design variables can be the cross-sectional areas A of individual bar members or the area moment of inertias I of the beam members. Configuration design variables are related to the orientations of components in built-up structures, such as the change shown in Figure 18.2(b), in which orientation angles of individual members are altered in addition to length changes. Shape design variables describe

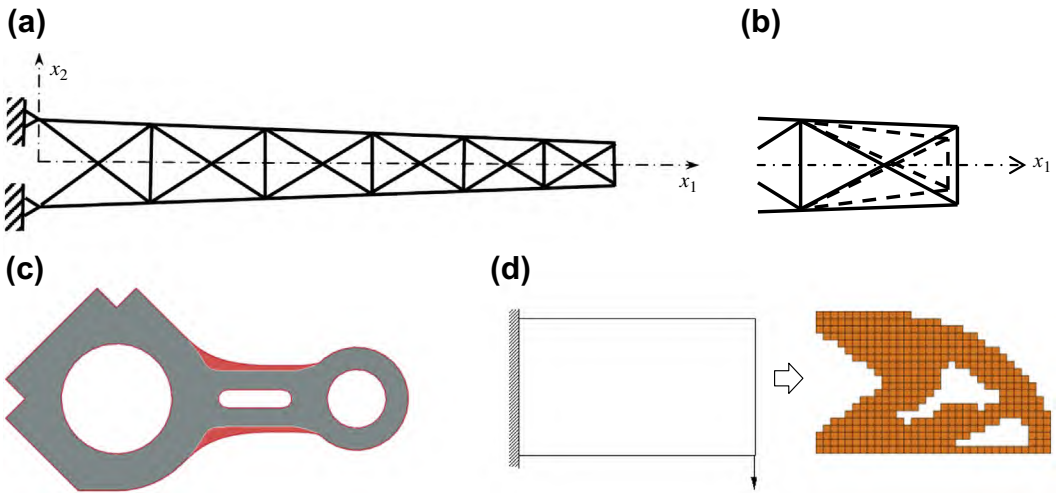


FIGURE 18.2

Illustration of design variables in different categories. (a) A built-up structure in which material parameters and cross-sectional areas of the bar members can be changed. (b) Configuration design by adjusting the orientations and lengths of truss members (Twu and Choi, 1993). (c) Shape design for a 2-D engine connecting rod (Edke and Chang, 2011). (d) Topology optimization of a solid beam (Tang and Chang, 2001).

the change in the length of one-dimensional (1-D) structures or the geometric shape of two-dimensional (2-D) and three-dimensional (3-D) structures, as illustrated in Figure 18.2(c). Topology optimization determines the layout of the structure, as depicted in Figure 18.2(d).

To carry out sensitivity analysis, it is obvious that the performance measure is presumed to be a differentiable function of the design, at least in the neighborhood of the current design. In general for structural designs, essential structural responses, such as displacement, stress, buckling load, frequency, and so forth, are differentiable functions with respect to design.

Sensitivity analysis for performance measures expressed explicitly in design variables, such as in Eq. 18.1, is straightforward. One derivative leads to sensitivity equations such as those in Eqs 18.2a and 18.2b. No expensive computation is required.

However, in most cases, performance measures cannot be expressed explicitly in design variables. In fact, in many cases, function evaluations must be carried out numerically using FEA. In these cases, there is no equation to take derivatives for. So, how do we calculate gradients to explore viable design alternatives and support batch mode design optimization?

In this chapter, we discuss sensitivity analysis for sizing, material, and shape designs, as well as topology optimization. We use simple and analytical examples to illustrate the sensitivity analysis concept and detailed solution techniques. We start by introducing the basic formulation of a simple bar structure in Section 18.2, which serves as an example problem to facilitate our follow-up discussion. We then provide an overview on sensitivity analysis methods in Section 18.3. In Section 18.4, we discuss sensitivity analysis for sizing and material design variables, especially using the continuum formulation. In Section 18.5, we discuss shape DSA, in which design parameterization and design velocity field computations are first discussed. Shape DSA methods are followed. In Section 18.6, we discuss topology optimization, which is effective for generating structural layouts in support of concept design. We offer one case study in Section 18.7 to showcase some of the typical design applications in practice. We assume that the performance measures cannot be expressed explicitly in design. Therefore, although simple problems are employed for illustration, readers should be mindful that in practice function evaluations are carried out numerically, such as using FEA.

18.2 SIMPLE BAR EXAMPLE

In this section, we introduce a basic concept in structural analysis that is essential for DSA. We use a simple bar example to illustrate the concept for the purpose of avoiding complex mathematical formulations that would dilute our focus. The bar example and formulation discussed in this section will be used in the next section to illustrate some of the solution techniques for sensitivity analysis.

We start by deriving a governing equation in the differential form with which we are familiar. Then, in Section 18.2.2, we introduce the energy equation that is equivalent to the differential governing equation but is more relaxed in the smoothness requirement on the solution function. The energy form is more general and is suitable to serve as the starting point when formulating equations for both FEA and sensitivity analysis. We use the simple bar example in Section 18.2.3 to illustrate the finite element formulation.

18.2.1 DIFFERENTIAL EQUATION

A 1-D bar under uniformly distributed load f (force per length) is shown in Figure 18.3(a). The bar has a uniform cross-sectional area A and length ℓ. The modulus of elasticity is E. We first formulate the differential equation that governs the structural response of the bar, and then we present the energy-based formulation for the same structure. Thereafter, we introduce the finite element equations of the problem derived from the energy formulation.

The governing differential equation of the bar being stretched by the distributed force f can be obtained from the free-body diagram in Figure 18.3(b), in which an arbitrary small element is cut off with forces acting on it. Note that $F(x + \Delta x)$ and $F(x)$ are internal forces acting on the small element and $f\Delta x$ is the distributed load acting on the small element. The internal force $F(x + \Delta x)$ can be written in a Taylor series expansion as

$$F(x + \Delta x) = F(x) + F(x)_{,1}\Delta x + \text{ higher order term} \tag{18.2}$$

in which the shorthand notation $F(x)_{,1} = \partial F(x)/\partial x$ is employed. The equilibrium equation of the small element can be formulated as

$$F(x + \Delta x) - F(x) + f\Delta x = 0 \tag{18.3}$$

Hence,

$$-F(x)_{,1}\Delta x + \text{higher order term} = f\Delta x \tag{18.4}$$

Dividing both sides of Eq. 18.4 by Δx and setting Δx to zero, we have

$$-F(x)_{,1} = f \tag{18.5}$$

Note that force $F(x)$ is stress σ times area A, and stress is the modulus of elasticity E times strain $\varepsilon = z_{,1}$, in which the shorthand notation $z_{,1} = \partial z/\partial x$ is employed again. Here, z is the axial displacement of the bar due to the distributed load f. Therefore, stress is $\sigma = Ez_{,1}$. Hence, Eq. 18.5 becomes

$$-\left(EAz_{,1}\right)_{,1} = f \tag{18.6a}$$

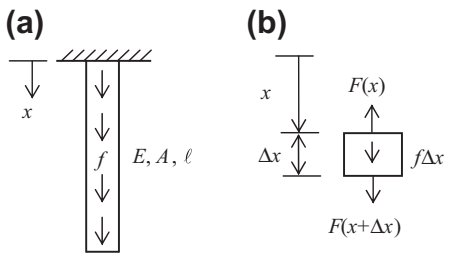


FIGURE 18.3

A one-dimensional bar example under static load. (a) Schematic view. (b) The forces acting on an arbitrary small element Δx.

Boundary conditions of the bar can be written as

$$z(0) = 0, \quad \text{and} \tag{18.6b}$$

$$EAz_{,1}(\ell) = 0 \tag{18.6c}$$

Displacement at the top ($x = 0$) is fixed and there is no traction force applied at the bottom end of the bar ($x = \ell$); hence, stress at the end times area is equal to zero. Note that Eq. 18.6b is called the essential boundary condition, and Eq. 18.6c is the natural boundary condition.

If the distributed load f is assumed to be a constant, then Eq. 18.6 implies that the derivative of E, A, and $z_{,1}$ with respect to x must exist and be at least to be a constant because the distributed force f is a constant. Mathematically, we write $E \in C^1[0, \ell]$ and $A \in C^1[0, \ell]$, in which $C^1[0, \ell]$ stands for the first-order derivative continuous in domain $[0, \ell]$. Hence, the second derivative of the solution z must exist and be at least a constant (or continuous). Therefore, the solution space of the differential equation of Eq. 18.6a and boundary conditions in Eq. 18.6b can be defined as

$$\overline{S} = \{z \in C^2[0, \ell] | z(0) = 0\} \tag{18.7}$$

in which we include only the essential boundary condition $z(0) = 0$, and $C^2[0, \ell]$ requires functions with the second derivative continuous in $[0, \ell]$. The governing equation in the differential form stated in Eqs 18.6a, 18.6b, and 18.6c is often called the "strong form" due to the high smoothness requirement of the solution function z—in this case, C^2.

Solving the differential equation gives the solution to the structural problem as

$$z^{\text{Exact}}(x) = -\frac{f}{2EA}x^2 + \frac{f\ell}{EA}x \tag{18.8}$$

If we are interested in designing the bar to reduce the displacement at the bottom end of $z(\ell)$, a displacement performance measure can be defined as

$$\psi = z(\ell) = \frac{f\ell^2}{2EA} \tag{18.9}$$

in which the displacement can be altered by changing material design variable E, sizing design variable A (cross-sectional area), or shape design variable ℓ (length of the bar).

18.2.2 ENERGY EQUATION

In practice, only a handful of structural problems can be formulated and solved analytically. In general, structural problems are solved by using the finite element method (FEM). The formulation of FEM usually starts from an energy equation (or variational equation), as will be illustrated next.

We derive the energy equation of the structural governing equation by multiplying both sides of Eq. 18.6a with an arbitrary virtual displacement $\overline{z}$ and then integrating over the structural domain $[0, \ell]$:

$$\int_0^\ell \left(-EAz_{,1}\right)_{,1} \overline{z} \, dx = \int_0^\ell f \, \overline{z} \, dx \tag{18.10}$$

Carrying out integration in part on Eq. 18.10 one time, we have

$$-\left(EAz_{,1}\overline{z}\right)\Big|_0^{\ell} + \int_0^{\ell} EAz_{,1}\overline{z}_{,1}\mathrm{d}x = \int_0^{\ell} f\,\overline{z}\mathrm{d}x \tag{18.11}$$

Both z and $\overline{z}$ belong to the space of kinematically admissible virtual displacement,

$$S = \left\{z \in H^1[0,\ell] \middle| z(0) = 0\right\} \tag{18.12}$$

where $z(0) = 0$ is the essential (or kinematic) boundary condition and $H^1[0, \ell]$ is the first-order Sobolev space. Because the boundary terms in Eq. 18.7 can be eliminated by considering $\overline{z}(0) = 0$ (essential boundary condition) and applying the natural boundary condition $z_{,1}(\ell) = 0$, the following energy equation is obtained for the bar problem:

$$a(z,\overline{z}) = \int_0^{\ell} EAz_{,1}\overline{z}_{,1}\,\mathrm{d}x = \int_0^{\ell} f\,\overline{z}\,\mathrm{d}x = \ell(\overline{z}) \tag{18.13}$$

which holds for all $\overline{z} \in S$ (or $\forall \overline{z} \in S$). Note that $a(z,\overline{z})$ and $\ell(\overline{z})$ are called the energy bilinear form and load linear form, respectively, thus implying that $a(z,\overline{z})$ is linear in both z and $\overline{z}$ while $\ell(\overline{z})$ is linear in $\overline{z}$.

If we replace the virtual displacement $\overline{z}$ with the physical displacement z for the energy bilinear form in Eq. 18.13, then

$$a(z,z) = \int_0^{\ell} A\left(Ez_{,1}\right)z_{,1}\mathrm{d}x = \int_0^{\ell} A\sigma\varepsilon\mathrm{d}x = 2U \tag{18.14}$$

U is the strain energy defined as

$$U = \int_{\Omega} u\mathrm{d}V = \int_0^{\ell}\left(\frac{1}{2}\sigma\varepsilon\right)A\mathrm{d}x = \frac{1}{2}a(z,z) \tag{18.15}$$

where u is the strain energy per volume V. If we do the same for the load linear form, then we have

$$\ell(z) = \int_0^{\ell} f\,z\mathrm{d}x \tag{18.16}$$

which is the work done by the external force f. Therefore, Eq. 18.13 is nothing but the equilibrium of work and energy in virtual displacement $\overline{z}$, which is formally called the principle of virtual work. The principle of virtual work states that, at equilibrium, the strain energy change due to a small virtual displacement is equal to the work done by the forces in moving through the virtual displacement. More specifically, the virtual displacement $\overline{z}$ is a small imaginary change in configuration that is also an admissible displacement and satisfies the essential boundary condition.

From a structural analysis perspective, the principle of virtual work states that the solution z is a function in the trial function space, which satisfies Eq. 18.13 for all virtual displacement $\overline{z}$ in the test function space. Note that, in general, the trial function space and test function space are identical for problems that involve only homogeneous essential boundary conditions; both are the kinematically admissible virtual displacement space S defined in Eq. 18.12. Homogeneous boundary conditions are either zero displacements or zero slope at the boundary. Note that the solution z is called the trial function, and the virtual displacement $\overline{z}$ is called the test function.

Note that solutions z in $H^1[0, \ell]$ imply that the virtual energy terms in Eq. 18.13—both $a(z, \overline{z})$ and $\ell(\overline{z})$—are finite; that is, $a(z, \overline{z}) < \infty$ and $\ell(\overline{z}) < \infty$. It is obvious that as long as $z_{,1}$ and $\overline{z}_{,1}$ exist and are integrable, not even being continuous, the integrals of $a(z, \overline{z})$ and $\ell(\overline{z})$ are finite. The requirements for the smoothness of the solution functions are significantly relaxed from that of space $\overline{S}$ defined in Eq. 18.7. For structural problems, work and energy are finite for a given external load that is finite, as seen in practice. The energy equation of Eq. 18.13 is also called the generalized formulation or "weak form" because the requirements for the solution functions are not as strong as those of differential equations. Without going through mathematic arguments, we state that the solution z of Eq. 18.13 exists and is unique. Rigourous arguments and proofs can be found in Haug et al. (1986).

18.2.3 FINITE ELEMENT FORMULATION

Consider discretizing the bar using two truss elements, each with length $\ell/2$, as shown in Figure 18.4. In this example, linear interpolation is used to describe the displacement field between nodal points, as follows:

$$z(x) = \begin{cases} \begin{bmatrix} 1 - \dfrac{2x}{\ell} & \dfrac{2x}{\ell} \end{bmatrix} \cdot \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}, & 0 \le x \le \ell/2 \\ \begin{bmatrix} 1 - \dfrac{2(x-\ell/2)}{\ell} & \dfrac{2(x-\ell/2)}{\ell} \end{bmatrix} \cdot \begin{bmatrix} Z_2 \\ Z_3 \end{bmatrix}, & \ell/2 < x \le \ell \end{cases} \tag{18.17}$$

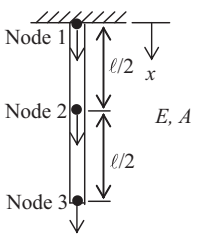


FIGURE 18.4

Finite element model of the 1-D bar structure. Two truss finite elements are used to discretize the structural domain.

where the row vectors contain the shape functions of individual elements, and Z_1, Z_2, and Z_3 represent nodal displacements at nodes 1, 2, and 3, respectively. Note that the virtual displacement $\overline{z}$ can be interpolated in the same way as the actual displacement z shown in Eq. 18.17. For clarification, lower-case z represents displacement function, and capital Z represents the nodal displacement obtained from FEA.

Discretizing the left- and right-hand sides of the energy equation (Eq. 18.13) using shape functions gives

$$\begin{aligned}
&\int_0^{\ell} EAz_{,1}\overline{z}_{,1}\mathrm{d}x = \int_0^{\ell/2} EAz_{,1}\overline{z}_{,1}\mathrm{d}x + \int_{\ell/2}^{\ell} EAz_{,1}\overline{z}_{,1}\mathrm{d}x \\
&\overset{\text{discretize}}{=} \begin{bmatrix}\overline{Z}_1\\ \overline{Z}_2\end{bmatrix}^{\mathrm{T}} \int_0^{\ell/2} EA \begin{bmatrix}-\frac{2}{\ell}\\ \frac{2}{\ell}\end{bmatrix}\begin{bmatrix}-\frac{2}{\ell} & \frac{2}{\ell}\end{bmatrix}\mathrm{d}x \begin{bmatrix}Z_1\\ Z_2\end{bmatrix} + \begin{bmatrix}\overline{Z}_2\\ \overline{Z}_3\end{bmatrix}^{\mathrm{T}} \int_{\ell/2}^{\ell} EA \begin{bmatrix}-\frac{2}{\ell}\\ \frac{2}{\ell}\end{bmatrix}\begin{bmatrix}-\frac{2}{\ell} & \frac{2}{\ell}\end{bmatrix}\mathrm{d}x \begin{bmatrix}Z_2\\ Z_3\end{bmatrix} \\
&= \begin{bmatrix}\overline{Z}_1\\ \overline{Z}_2\end{bmatrix}^{\mathrm{T}} \frac{2EA}{\ell}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\begin{bmatrix}Z_1\\ Z_2\end{bmatrix} + \begin{bmatrix}\overline{Z}_2\\ \overline{Z}_3\end{bmatrix}^{\mathrm{T}} \frac{2EA}{\ell}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\begin{bmatrix}Z_2\\ Z_3\end{bmatrix} \\
&= \begin{bmatrix}\overline{Z}_1\\ \overline{Z}_2\\ \overline{Z}_3\end{bmatrix}^{\mathrm{T}} \cdot \frac{2EA}{\ell}\begin{bmatrix}1 & -1 & 0\\ -1 & 2 & -1\\ 0 & -1 & 1\end{bmatrix}\cdot\begin{bmatrix}Z_1\\ Z_2\\ Z_3\end{bmatrix} = \overline{\mathbf{Z}}_{\mathrm{g}}\mathbf{K}_{\mathrm{g}}\mathbf{Z}_{\mathrm{g}}
\end{aligned} \tag{18.18}$$

and

$$\begin{aligned}
&\int_0^{\ell} f\,\overline{z}\mathrm{d}x = \int_0^{\ell/2} f\,\overline{z}\mathrm{d}x + \int_{\ell/2}^{\ell} f\,\overline{z}\mathrm{d}x \\
&\overset{\text{discretize}}{=} \begin{bmatrix}\overline{Z}_1\\ \overline{Z}_2\end{bmatrix}^{\mathrm{T}} \int_0^{\ell/2} f \begin{bmatrix}1-\frac{2x}{\ell}\\ \frac{2x}{\ell}\end{bmatrix}\mathrm{d}x + \begin{bmatrix}\overline{Z}_2\\ \overline{Z}_3\end{bmatrix}^{\mathrm{T}} \int_{\ell/2}^{\ell} f \begin{bmatrix}1-\frac{2(x-\ell/2)}{\ell}\\ \frac{2(x-\ell/2)}{\ell}\end{bmatrix}\mathrm{d}x \\
&= \begin{bmatrix}\overline{Z}_1\\ \overline{Z}_2\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}\frac{\ell}{4}\\ \frac{\ell}{4}\end{bmatrix}\mathrm{d}x + \begin{bmatrix}\overline{Z}_2\\ \overline{Z}_3\end{bmatrix}^{\mathrm{T}} \begin{bmatrix}\frac{\ell}{4}\\ \frac{\ell}{4}\end{bmatrix} = \begin{bmatrix}\overline{Z}_1\\ \overline{Z}_2\\ \overline{Z}_3\end{bmatrix}^{\mathrm{T}} \cdot \begin{bmatrix}f\ell/4\\ f\ell/2\\ f\ell/4\end{bmatrix} = \overline{\mathbf{Z}}_{\mathrm{g}}\mathbf{F}_{\mathrm{g}}
\end{aligned} \tag{18.19}$$

where $\mathbf{K}_g$ is the generalized global stiffness matrix. $\mathbf{Z}_g$, $\overline{\mathbf{Z}}_g$, and $\mathbf{F}_g$ are the global displacement, virtual displacement, and force vectors, respectively. Note that the distributed load f has been converted into point loads acting upon the nodal points due to discretization.

Because the left-hand sides of Eqs 18.18 and 18.19 are equal to each other, the global finite element matrix equation can be obtained by eliminating the arbitrary virtual displacement $\overline{\mathbf{Z}}_g$, as follows:

$$\mathbf{K}_g\mathbf{Z}_g = \frac{2EA}{\ell}\begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \end{bmatrix} = \begin{bmatrix} f\ell/4 \\ f\ell/2 \\ f\ell/4 \end{bmatrix} = \mathbf{F}_g \tag{18.20}$$

This equation cannot be solved due to the singularity of $\mathbf{K}_g$. By imposing the essential boundary condition $z(0) = 0$, we can remove Z_1 from Eq. 18.20, giving

$$\mathbf{KZ} = \frac{2EA}{\ell}\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} Z_2 \\ Z_3 \end{bmatrix} = \begin{bmatrix} f\ell/2 \\ f\ell/4 \end{bmatrix} = \mathbf{F} \tag{18.21}$$

where

$$\mathbf{K} = \frac{2EA}{\ell}\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \tag{18.22}$$

This is called the reduced global stiffness matrix, which is nonsingular (or more precisely, positive definite). Solving the reduced matrix equation (Eq. 18.22) by inversing the stiffness matrix gives the nodal displacement solution:

$$\mathbf{Z} = \mathbf{K}^{-1}\mathrm{F} = \frac{\ell}{2EA}\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} f\ell/2 \\ f\ell/4 \end{bmatrix} = \begin{bmatrix} \dfrac{3f\ell^2}{8EA} \\ \\ \dfrac{f\ell^2}{2EA} \end{bmatrix} \tag{18.23}$$

Based on the nodal point solutions given above, the displacement at an arbitrary location in the domain $[0, \ell]$ can be interpolated using the shape functions defined in Eq. 18.17, as follows:

$$\mathbf{z}(x) = \begin{cases} \begin{bmatrix} 1 - \dfrac{2x}{\ell} & \dfrac{2x}{\ell} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \\ \dfrac{3f\ell^2}{8EA} \end{bmatrix} = \dfrac{2x}{\ell}\dfrac{3f\ell^2}{8EA}, & 0 \le x \le \ell/2 \\ \\ \begin{bmatrix} 1 - \dfrac{2(x-\ell/2)}{\ell} & \dfrac{2(x-\ell/2)}{\ell} \end{bmatrix} \cdot \begin{bmatrix} \dfrac{3f\ell^2}{8EA} \\ \\ \dfrac{f\ell^2}{2EA} \end{bmatrix} = \left(1 - \dfrac{2(x-\ell/2)}{\ell}\right)\dfrac{3f\ell^2}{8EA} + \dfrac{2(x-\ell/2)}{\ell}\dfrac{f\ell^2}{2EA}, & \ell/2 < x \le \ell \end{cases} \tag{18.24}$$

Note that in computer implementation, the integration of Eqs 18.18 and 18.19 is carried out numerically (e.g., using Gauss integration), and the stiffness matrix **K** is decomposed or factorized numerically (e.g., using Lower Upper (LU) decomposition; Atkinson, 1989) for solving the displacement vector **Z**, instead of fully inversed as stated in Eq. 18.23.

It is of note that the finite element displacement solution $\mathbf{z}(x)$ is piecewise linear (linear in individual elements); however, the exact solution obtained by solving the differential governing equation is a quadratic function of x (Eq. 18.8). They are not identical. This is because linear shape functions are used for interpolation finite element solution in Eq. 18.17, whereas the analytical solution to the problem is a quadratic function of x. However, the finite element solution matches the exact solution only at nodal points; that is, $Z_2 = z(\ell/2)$ and $Z_3 = z(\ell)$.

The stress in the bar can be calculated using the exact solution (Eq. 18.8) as

$$\sigma^{\text{Exact}} = E\left[z^{\text{Exact}}(x)\right]_{,1} = -\frac{f}{A}x + \frac{f\ell}{A} = \frac{f}{A}(\ell - x) \tag{18.25}$$

which is always greater than or equal to zero, implying that the bar is stretched due to the distributed load f.

Using the finite element solution of Eq. 18.24, stress in the bar is calculated as

$$\sigma(x) = Ez_{,1}(x) = \begin{cases} E\left[-\frac{2}{\ell} \quad \frac{2}{\ell}\right] \cdot \begin{bmatrix} 0 \\ \frac{3f\ell^2}{8EA} \end{bmatrix} = \frac{3f\ell}{4A}, & 0 \le x \le \ell/2 \\ E\left[-\frac{2}{\ell} \quad \frac{2}{\ell}\right] \cdot \begin{bmatrix} \frac{3f\ell^2}{8EA} \\ \frac{f\ell^2}{2EA} \end{bmatrix} = \frac{f\ell}{4A}, & \ell/2 < x \le \ell \end{cases} \tag{18.26}$$

which is a piecewise constant function. It is obvious that the stress solved by FEA is not continuous across finite elements. In this bar example, stress at node 2 is not continuous, which is referred to as stress jump, whereas the analytical stress is a linear function of x, as shown in Eq. 18.25. Stress jump is physically impossible. This piecewise constant stress is due to the fact that we use linear shape functions to interpolate displacement results. An important point to note is that, as the finite element mesh is refined (element size reduced), a better solution revealing less stress jump is expected.

18.3 SENSITIVITY ANALYSIS METHODS

In general, structural performance measures are categorized as global and local measures. Global measures, such as volume or weight, vibration frequency, compliance, and buckling load, are measured for the entire structure. On the other hand, measures such as stresses or displacements are defined at specific points or small areas (e.g., average stress in a finite element). Design variables in general include shape and nonshape. Nonshape design variables do not alter the geometric shape of the structure, such as sizing and material. Design variables that alter the geometric shape of the structure include domain shape and configuration of frame structures. In addition, topology optimization considers element density as design variables in general (further discussed in Section 18.6).

Four approaches can be employed for DSA: the analytical derivative, overall finite difference, discrete approach, and continuum approach. As stated at the beginning of Section 18.2, sensitivity analysis calculates the gradient or sensitivity of the performance measure(s) ψ with respect to design variable b. For simplicity, we assume a single performance measure and a single design variable for the time being.

The analytical derivative method (or analytical method) calculates the sensitivity coefficients by taking the derivative of the measure with respect to the design variable analytically. This method requires explicit expression of the measure in terms of the design variable. In practice, only a handful of measures can be expressed in the design variable explicitly. Therefore, the analytical method is very limited in practical applications.

Using the overall finite difference method, sensitivity coefficients are obtained by rerunning structural analysis at a perturbed design and calculating the difference of the measure at the current and perturbed designs. The sensitivity coefficient is then approximated by dividing the difference in performance measure values by the design perturbation.

In the discrete approach, the sensitivity formulation is obtained by taking derivatives of the finite element matrix equations. If the derivatives of the finite element equations are obtained analytically, it is called the discrete-analytical method, in which discrete refers to its formulation, and analytical refers to the nature of the derivatives. If the derivatives are obtained using finite differences, then the method is called a discrete–discrete or semi-analytical method. The semi-analytical method is probably the most employed approach other than the overall finite difference in practice. Although both the overall finite difference and semianalytical methods are very general, they require perturbing the design variable(s) by a small amount. An adequate design perturbation depends on the characteristics of the function and is sometimes difficult to acquire.

Another approach for sensitivity analysis is called the continuum approach, in which the sensitivity formulation is obtained by taking derivatives of the energy equations in the integral or continuum form, instead of the finite element equations that are discretized—hence the name of the approach. The derivatives of the energy equations are then discretized in finite element formulation and solved by FEA; such a method is called continuum-discrete. In some rare cases, the solutions of the structural problems are obtained analytically without an FEA. These analytical solutions can then be plugged into the continuum sensitivity expression without discretization; this is called the continuum-analytical method. However, because the analytical solution of structural problems is rare, the applications of the continuum-analytical method are limited. One important feature of the continuum approach is that neither the continuum-discrete or continuum-analytical method requires a design perturbation; they are superior to the overall finite difference and semi-analytical methods. Also, in many structural design problems, the continuum-discrete method is equivalent to the discrete-analytical method.

The sensitivity analysis methods mentioned above are listed in Figure 18.5, in which we assume a linear elastic problem under static load. Problems other than static may see a slightly different classification than that in Figure 18.5.

The sensitivity analysis methods listed above will be further discussed next. Note that, in practice, the continuum-discrete approach is considered the best because it is general, accurate, and efficient. More importantly, this method can be implemented outside commercial FEA codes that are employed for analysis. In the following, we discuss individual methods and point out their pros and cons. We use the simple bar example discussed in Section 18.2 to illustrate the detailed derivations.

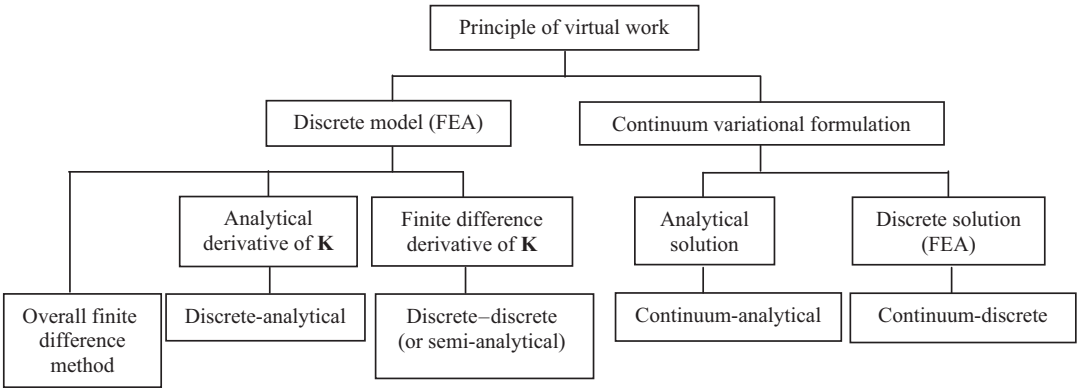


FIGURE 18.5

Major approaches for DSA.

18.3.1 ANALYTICAL DERIVATIVE METHOD

The analytical derivative method (or analytical method) is straightforward. For the simple bar example, the displacement is obtained by solving the differential equation in Section 18.2.1 as

$$z(x) = -\frac{f}{2EA}x^2 + \frac{f\ell}{EA}x \tag{18.27}$$

If we are interested in designing the bar with a reduced displacement at its bottom end, a displacement performance measure can be defined as

$$\psi = z(\ell) = \frac{f\ell^2}{2EA} \tag{18.28}$$

If the cross-sectional area A is the design variable, the sensitivity can be calculated by taking derivative of the measure with respect to the design variable analytically:

$$\frac{\mathrm{d}\psi}{\mathrm{d}A} = -\frac{f\ell^2}{2EA^2} \tag{18.29}$$

The negative sensitivity coefficient implies that increasing the cross-sectional area A decreases the displacement at the bottom end of the bar.

Next, we consider the stress in the bar as the performance measure, which can be calculated, as in Eq. 18.25, as

$$\sigma^{\text{Exact}} = E\left(z^{\text{Exact}}\right)_{,1}(x) = -\frac{f}{A}x + \frac{f\ell}{A} \tag{18.30}$$

We define a stress measure at the root of the bar ($x = 0$):

$$\psi = \frac{f\ell}{A} \tag{18.31}$$

The sensitivity of stress with the area design variable is

$$\frac{d\psi}{dA} = -\frac{f\ell}{A^2} \tag{18.32}$$

This equation shows that increasing the cross-sectional area A decreases the stress at the root of the bar. Another frequently used stress measure is the average stress, which is defined as

$$\psi = \sigma_{ave} = \frac{\int_0^\ell \sigma dx}{\ell} = \frac{f\ell}{2A} \tag{18.33}$$

The sensitivity of the average stress can be calculated as

$$\frac{d\psi}{dA} = -\frac{f\ell}{2A^2} \tag{18.34}$$

The discussion above assumes analytical expression of measure ψ in design variable. In general, in rare cases such an expression exists.

18.3.2 OVERALL FINITE DIFFERENCE

The easiest way to compute the sensitivity information of a performance measure without analytical solutions is by using the overall finite difference method. The overall finite difference method computes the sensitivity by evaluating performance measures at different values of design variables, i.e., at perturbed designs. Although the given design problem may have many design variables, a single design variable is perturbed at a time. We assume one design variable in the following explanation. If b is the current design, then the analysis results provide the value of performance measure $\psi(b)$—for example, using FEA.

In addition, if the design is perturbed to $b + \Delta b$, where Δb represents a small change or perturbation in the design variable, then the sensitivity of $\psi(b)$ can be approximated as

$$\frac{\partial\psi}{\partial b} \approx \frac{\psi(b + \Delta b) - \psi(b)}{\Delta b} \tag{18.35}$$

This is called the forward difference method because the design is perturbed by $+\Delta b$. If $-\Delta b$ is substituted in Eq. 18.35 for $+\Delta b$, then the equation is defined as the backward difference method. Additionally, if the design is perturbed in both directions, such that the design sensitivity is approximated by

$$\frac{\partial\psi}{\partial b} \approx \frac{\psi(b + \Delta b) - \psi(b - \Delta b)}{2\Delta b} \tag{18.36}$$

then the equation is defined as the central difference method. The advantage of the finite difference method is obvious. If structural analysis can be performed and the performance measure can be obtained as a result of structural analysis, then the expressions in Eqs 18.35 and 18.36 become virtually independent of the problem types considered. Consequently, this method is very general and has been widely employed in engineering designs beyond structural problems.

However, sensitivity computation costs become expensive. If n represents the number of design variables, then $n + 1$ number of analyses (including the current design b) have to be carried out for

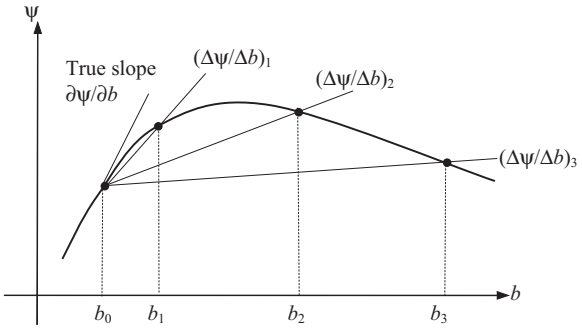


FIGURE 18.6

Influence of step size in the forward finite difference method.

either forward or backward differences, and $2n$ analyses are required for the central differences. This method, although easy to implement, may not be feasible for large-scale problems with many design variables due to extensive computational efforts.

Another major disadvantage of the finite difference method is the degree of accuracy of its sensitivity results. In Eqs 18.35 and 18.36, accurate results can be expected when Δb approaches zero. Figure 18.6 shows some sensitivity results using the finite difference method. The tangential slope of the curve at b_0—that is, $\frac{\partial \psi}{\partial b}\big|_{b=b_0}$—is the exact sensitivity value, which is also the true slope of the tangent line at b_0. Depending on the perturbation size, the sensitivity results are quite different; for example,

$$\left(\frac{\Delta\psi}{\Delta b}\right)_1 = \frac{\psi(b_1)-\psi(b_0)}{b_1-b_0}, \quad \left(\frac{\Delta\psi}{\Delta b}\right)_2 = \frac{\psi(b_2)-\psi(b_0)}{b_2-b_0}, \quad \text{and} \quad \left(\frac{\Delta\psi}{\Delta b}\right)_3 = \frac{\psi(b_3)-\psi(b_0)}{b_3-b_0}$$

For a mildly nonlinear performance measure, a relatively large perturbation can still provide a reasonable estimation of sensitivity results. However, for a highly nonlinear performance measure, even a small perturbation yields inaccurate results. Thus, the determination of perturbation size Δb greatly affects the accuracy of the sensitivity coefficients.

On the other hand, although it may be necessary to choose a very small perturbation, numerical noise becomes dominant for a perturbation size that is too small. With a too small perturbation, no reliable difference can be found in the numerical analysis results. Because the behavior of a performance measure in design space is unknown a priori, it is difficult to determine design perturbation sizes that work for all problems.

Usually, a convergence study in design perturbation size Δb is carried out first for a given problem, in which a relatively large perturbation size is assumed at the beginning. By reducing the perturbation size, if the finite difference values shown in Eqs 18.35 or 18.36 approach or converge to a specific value, then the true sensitivity, or the true slope of the tangent line at current design b_0, is accurately approximated. Theoretically, when the perturbation size Δb approaches zero, the finite difference value equals the true slope; that is,

$$\frac{\partial \psi}{\partial b} = \lim_{\Delta b \to 0} \frac{\psi(b+\Delta b)-\psi(b)}{\Delta b} \tag{18.37}$$

However, as discussed, the perturbation size cannot be so small as to induce numerical noise.

EXAMPLE 18.1

For a sinusoidal function $f(b) = \sin b$, calculate its sensitivity at $b = 0$. Use the forward finite difference method to approximate the sensitivity for $\Delta b = 1, 0.5, 0.1, 0.01$.

Solution

The derivative of $f(b) = \sin b$ with respect to b is

$$\frac{df}{db} = \cos b$$

which gives the analytical sensitivity coefficient at $b_0 = 0$, $df/db = 1$. Now, we use Eq. 18.35 to approximate the sensitivity of the function $f(b)$ for $\Delta b = 1, 0.5, 0.1, 0.01$.

$$\left(\frac{\Delta f}{\Delta b}\right)_{\Delta b=1} = \frac{\sin(1) - \sin(0)}{1} = \frac{0.8415 - 0}{1} = 0.8415$$

$$\left(\frac{\Delta f}{\Delta b}\right)_{\Delta b=0.5} = \frac{\sin(0.5) - \sin(0)}{0.5} = \frac{0.4794 - 0}{0.5} = 0.9589$$

$$\left(\frac{\Delta f}{\Delta b}\right)_{\Delta b=0.1} = \frac{\sin(0.1) - \sin(0)}{0.1} = \frac{0.09983 - 0}{0.1} = 0.9983$$

$$\left(\frac{\Delta f}{\Delta b}\right)_{\Delta b=0.01} = \frac{\sin(0.01) - \sin(0)}{0.01} = \frac{0.0099998 - 0}{0.01} = 0.99998$$

As shown above, reducing Δb leads to more accurate approximations of the sensitivity of $f(b)$. In fact, for the sine function, when Δb is 0.1, the error in the approximation using the forward finite difference method is less than 1% because the function is not highly nonlinear.

18.3.3 DISCRETE APPROACH

The sensitivity formulation of the discrete approach is obtained by taking derivatives of the discretized finite element equations. For a static problem, as discussed in the simple bar example in Section 18.2, we have

$$\mathbf{KZ} = \mathbf{F} \tag{18.38}$$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{F}$ is the vector of the external load, and $\mathbf{Z}$ is the nodal point displacement of the structure obtained by solving the matrix equations. Taking the derivatives of the finite element equations with respect to the design variable b yields

$$\mathbf{K}\frac{\partial \mathbf{Z}}{\partial b} + \frac{\partial \mathbf{K}}{\partial b}\mathbf{Z} = \frac{\partial \mathbf{F}}{\partial b} \tag{18.39}$$

Moving $\frac{\partial \mathbf{K}}{\partial b}\mathbf{Z}$ to the right-hand side, we have

$$\mathbf{K}\frac{\partial \mathbf{Z}}{\partial b} = \frac{\partial \mathbf{F}}{\partial b} - \frac{\partial \mathbf{K}}{\partial b}\mathbf{Z} \tag{18.40}$$

in which the terms on the right-hand side are a vector of so-called fictitious load. The derivative of the nodal displacements can be solved by treating the right-hand side of Eq. 18.40 as an additional load case:

$$\frac{\partial \mathbf{Z}}{\partial b} = \mathbf{K}^{-1}\left(\frac{\partial \mathbf{F}}{\partial b} - \frac{\partial \mathbf{K}}{\partial b}\mathbf{Z}\right) \tag{18.41}$$

in which the decomposed stiffness matrix used for solving $\mathbf{Z}$ in Eq. 18.38 can be used again to solve for $\partial\mathbf{Z}/\partial b$, which makes the approach more efficient than the overall finite difference method.

EXAMPLE 18.2

Calculate the sensitivity of the displacement at the bottom end of the simple bar example using the discrete-analytical approach, assuming the modulus of elasticity E as the design variable.

Solution

Recall the finite element equations of the bar example in Eq. 18.21:

$$\mathbf{KZ} = \frac{2EA}{\ell}\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} Z_2 \\ Z_3 \end{bmatrix} = \begin{bmatrix} f\ell/2 \\ f\ell/4 \end{bmatrix} = \mathbf{F} \tag{18.42}$$

Taking derivatives of the stiffness matrix $\mathbf{K}$ and load vector $\mathbf{F}$, we have

$$\frac{\partial \mathbf{K}}{\partial E} = \frac{2A}{\ell}\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \quad \text{and} \quad \frac{\partial \mathbf{F}}{\partial E} = \mathbf{0} \tag{18.43}$$

Therefore, from Eq. 18.41, we have

$$\frac{\partial \mathbf{Z}}{\partial E} = \begin{bmatrix} \frac{\partial Z_2}{\partial E} \\ \frac{\partial Z_3}{\partial E} \end{bmatrix} = \mathbf{K}^{-1}\left(\frac{\partial \mathbf{F}}{\partial E} - \frac{\partial \mathbf{K}}{\partial E}\mathbf{Z}\right) = \left(\frac{\ell}{2EA}\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}\right)\left(\mathbf{0} - \frac{2A}{\ell}\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} \frac{3f\ell^2}{8EA} \\ \\ \frac{f\ell^2}{2EA} \end{bmatrix}\right) = \begin{bmatrix} \frac{-3f\ell^2}{8E^2A} \\ \\ \frac{-f\ell^2}{2E^2A} \end{bmatrix} \tag{18.44}$$

Note that the second term in the result vector is $\frac{\partial Z_3}{\partial E} = \frac{\partial Z(\ell)}{\partial E} = -\frac{f\ell^2}{2E^2A}$, which can be also obtained by directly taking the derivative of $z(\ell) = -\frac{f\ell^2}{2EA}$ with respect to E.

Equation (18.41) represents a direct differentiation method of the discrete-analytical approach, in which the gradient is obtained analytically by taking the derivative of the finite element equations that are discretized.

For this simple example, the stiffness matrix and load vector are expressed in terms of the material design variable E explicitly. As a result, $\partial\mathbf{K}/\partial E$ and $\partial\mathbf{F}/\partial E$ can be obtained analytically. However, in general, such explicit expressions are not available. They are either too complex to formulate or entirely impossible. Especially when we use commercial FEA software, the stiffness matrix and force vector can only be retrieved as numerical data. Differentiating the matrix and vector in numerical data is out of the question. So, how do we calculate the gradients when we use commercial FEA software for structural analysis? How do we take derivatives of the stiffness matrix and force vector numerically? One possibility is to use finite differences, which can be stated mathematically as

$$\frac{\partial \mathbf{Z}}{\partial b} \approx \mathbf{K}^{-1}\left(\frac{\Delta \mathbf{F}}{\Delta b} - \frac{\Delta \mathbf{K}}{\Delta b}\mathbf{Z}\right) \tag{18.45}$$

in which

$$\frac{\Delta \mathbf{K}}{\Delta b} = \frac{\mathbf{K}(b + \Delta b) - \mathbf{K}(b)}{\Delta b} \tag{18.46}$$

and

$$\frac{\Delta \mathbf{F}}{\Delta b} = \frac{\mathbf{F}(b + \Delta b) - \mathbf{F}(b)}{\Delta b} \tag{18.47}$$

Note that in Eq. 18.46, $\mathbf{K}(b + \Delta b)$ can be obtained by perturbing the design variable b (e.g., the modulus of elasticity E) by a small amount Δb, and then asking the FEA code to generate the stiffness matrix at the perturbed design $b + \Delta b$. $\mathbf{F}(b + \Delta b)$ can be obtained in a similar way, although in Example 18.2, we know $\partial \mathbf{F}/\partial E = 0$; therefore, $\mathbf{F}(E + \Delta E)$ is not needed in this case.

Equations (18.45) to (18.47) represent the discrete–discrete or semi-analytical approach because the formulations are based on the discretized formulation in FEA, and the derivatives of the matrix and vector are obtained numerically using finite differences. Although this method is general, determining an adequate design perturbation Δb for accurate derivatives is not straightforward, as discussed in Section 18.3.2. Moreover, retrieving stiffness matrices and load vectors for the current and perturbed designs from commercial FEA codes and approximating their derivatives are nontrivial and sometimes inefficient for large-scale problems in implementation.

18.3.3.1 Direct Differentiation Method

Recall that Eq. 18.41 represents the direct differentiation method of the discrete-analytical approach. Equation (18.41) is good for displacement sensitivity. For structural design, performance measures are more than just displacements. In general, a performance measure ψ depends on the design explicitly and implicitly. That is, the performance measure ψ is in general a function of the design variable b and displacement (also called the state variable in the literature) $\mathbf{z}(b)$, as follows:

$$\psi = \psi(\mathbf{z}(b), b) \tag{18.48}$$

In general $\mathbf{z} = [z_1, z_2, z_3]^{\mathrm{T}}$, and $\mathbf{z} = z_1$ for one-dimensional problems. The sensitivity of function ψ can thus be expressed using the chain rule. Example 18.3 illustrates such a chain rule for differentiation.

EXAMPLE 18.3

A performance measure is defined as

$$\psi = \psi(z(b), b) = bz(b)^2 \tag{18.49}$$

Calculate the sensitivity of ψ with respect to the design variable b using chain rule.

Solution

Taking the derivative of ψ with respect to b, we have

$$\frac{\mathrm{d}\psi}{\mathrm{d}b} = \frac{\partial \psi}{\partial b} + \frac{\partial \psi}{\partial z}\frac{\partial z}{\partial b} = z^2 + 2bz\frac{\partial z}{\partial b} \tag{18.50}$$

Continued

EXAMPLE 18.3—cont'd

Note that the first term on the right-hand side of Eq. 18.50 shows the explicit dependence of ψ on b, which is the partial derivative of ψ with respect to b; that is, $\frac{\partial \psi}{\partial b} = z^2$. The second term consists of $\frac{\partial \psi}{\partial z} = \frac{\partial (bz^2)}{\partial z} = 2bz$, and $\frac{\partial z}{\partial b}$. Note that the second term quantifies the implicit dependence of ψ on b through $\frac{\partial z}{\partial b}$.

If $z = b^2$, then the performance measure becomes $\psi = bz(b)^2 = b^5$. Using Eq. 18.50, the sensitivity is

$$\frac{\mathrm{d}\psi}{\mathrm{d}b} = z^2 + 2bz\frac{\partial z}{\partial b} = b^4 + 2b\left(b^2\right)(2b) = 5b^4 \tag{18.51}$$

The same result can be obtained by taking the derivative directly for $\psi = b^5$. In a structural problem, the derivative of displacement $\mathbf{z}$ on the design variable b cannot be obtained by taking the derivative explicitly because an expression such as $z = b^2$ does not exist.

With the understanding of the chain rule shown in Example 18.3, we take the derivative of the performance measure stated in Eq. 18.48 with respect to b as

$$\frac{\mathrm{d}\psi}{\mathrm{d}b} = \frac{\partial \psi}{\partial b} + \frac{\partial \psi}{\partial \mathbf{z}}\frac{\partial \mathbf{z}}{\partial b} \tag{18.52a}$$

in which $\mathbf{z}$ is a continuous function of location $\mathbf{x}$ and design b, or

$$\frac{\mathrm{d}\psi}{\mathrm{d}b} = \frac{\partial \psi}{\partial b} + \frac{\partial \psi}{\partial \mathbf{Z}}\frac{\partial \mathbf{Z}}{\partial b} \tag{18.52b}$$

Here, $\mathbf{Z}$ represents the nodal point displacements obtained from FEA. Note that Eq. 18.52b is employed more often. The first term $\frac{\partial \psi}{\partial b}$ represents the explicit dependence of ψ on design variable b, and the second term shows the implicit dependence of ψ on design variable b through $\frac{\partial \mathbf{Z}}{\partial b}$. In structural problems, the calculation of $\frac{\partial \psi}{\partial b}$ is straightforward because the explicit dependence of ψ on design variable b is usually available. The second term in Eq. 18.52 consists of $\frac{\partial \psi}{\partial \mathbf{Z}}$ and $\frac{\partial \mathbf{Z}}{\partial b}$. $\frac{\partial \psi}{\partial \mathbf{Z}}$ can be calculated easily because measure ψ is usually explicitly defined in terms of displacement $\mathbf{Z}$. The term that shows the actual implicit dependence of the performance measure ψ on design variable b is $\frac{\partial \mathbf{Z}}{\partial b}$, which can be calculated in two different ways; both involve FEA for additional load cases using the decomposed stiffness matrix $\mathbf{K}$. One approach is the direct differentiation method, in which Eq. 18.41 or 18.45 is used to solve $\frac{\partial \mathbf{Z}}{\partial b}$.

Once $\partial \mathbf{Z}/\partial b$ is calculated, the sensitivity of a performance measure ψ, defined as a function of (discretized) state variable $\mathbf{Z}$, can be obtained using the direct differentiation method as

$$\frac{\mathrm{d}\psi}{\mathrm{d}b} = \frac{\partial \psi}{\partial b} + \frac{\partial \psi}{\partial \mathbf{Z}}\mathbf{K}^{-1}\left(\frac{\partial \mathbf{F}}{\partial b} - \frac{\partial \mathbf{K}}{\partial b}\mathbf{Z}\right) \tag{18.53}$$

EXAMPLE 18.4

We define the stress of the second element (with ends of node 2 and node 3) of the simple bar shown in Figure 18.4 as a performance measure. As shown in Eq. 18.26, the element stress is defined in terms of nodal point displacement $\mathbf{Z} = [Z_2, Z_3]^{\mathrm{T}}$ as

$$\psi = \sigma = Ez_{,1} = E\left[1 - \frac{2(x-\ell/2)}{\ell} \quad \frac{2(x-\ell/2)}{\ell}\right]_{,1} \cdot \begin{bmatrix} Z_2 \\ Z_3 \end{bmatrix} = E\left[-\frac{2}{\ell} \quad \frac{2}{\ell}\right] \cdot \begin{bmatrix} Z_2 \\ Z_3 \end{bmatrix} \tag{18.54}$$

EXAMPLE 18.4—cont'd

in which the dependence of ψ in design E is both explicit and implicit (through $\mathbf{Z}$). Assuming modulus E and area A as the design variables, calculate the sensitivity coefficients of the stress measure ψ using the direct differentiation method of the discrete-analytical approach.

Solution

For modulus design variable E, the sensitivity of the stress measure ψ can be calculated using Eq. 18.53, as follows:

$$\frac{\mathrm{d}\psi}{\mathrm{d}E}=\frac{\partial\psi}{\partial E}+\frac{\partial\psi}{\partial\mathbf{Z}}\mathbf{K}^{-1}\left(\frac{\partial\mathbf{F}}{\partial E}-\frac{\partial\mathbf{K}}{\partial E}\mathbf{Z}\right)=\begin{bmatrix}-\frac{2}{\ell} & \frac{2}{\ell}\end{bmatrix}\cdot\begin{bmatrix}Z_2\\ Z_3\end{bmatrix}+\begin{bmatrix}\frac{\partial\psi}{\partial Z_2} & \frac{\partial\psi}{\partial Z_3}\end{bmatrix}\mathbf{K}^{-1}\left(\frac{\partial\mathbf{F}}{\partial E}-\frac{\partial\mathbf{K}}{\partial E}\mathbf{Z}\right)$$

$$=\begin{bmatrix}-\frac{2}{\ell} & \frac{2}{\ell}\end{bmatrix}\cdot\begin{bmatrix}\frac{3f\ell^2}{8EA}\\ \frac{f\ell^2}{2EA}\end{bmatrix}+\begin{bmatrix}-\frac{2E}{\ell} & \frac{2E}{\ell}\end{bmatrix}\left(\frac{\ell}{2EA}\begin{bmatrix}1 & 1\\ 1 & 2\end{bmatrix}\right)\left(\mathbf{0}-\frac{2A}{\ell}\begin{bmatrix}2 & -1\\ -1 & 1\end{bmatrix}\begin{bmatrix}\frac{3f\ell^2}{8EA}\\ \frac{f\ell^2}{2EA}\end{bmatrix}\right) \tag{18.55}$$

$$=\begin{bmatrix}-\frac{2}{\ell} & \frac{2}{\ell}\end{bmatrix}\cdot\begin{bmatrix}\frac{3f\ell^2}{8EA}\\ \frac{f\ell^2}{2EA}\end{bmatrix}+\begin{bmatrix}-\frac{2E}{\ell} & \frac{2E}{\ell}\end{bmatrix}\begin{bmatrix}-\frac{3f\ell^2}{8E^2A}\\ -\frac{f\ell^2}{2E^2A}\end{bmatrix}=\frac{f\ell}{4EA}-\frac{f\ell}{4EA}=0$$

in which Eq. 18.54 is used to calculate the explicit dependent term $\frac{\partial\psi}{\partial E}$, which can be obtained easily once the solution $\mathbf{Z}$ is available. In this example, the explicit dependence is $\frac{\partial\psi}{\partial E}=\frac{f\ell}{4EA}$. The implicit term is $\frac{\partial\psi}{\partial\mathbf{Z}}\mathbf{K}^{-1}\left(\frac{\partial\mathbf{F}}{\partial E}-\frac{\partial\mathbf{K}}{\partial E}\mathbf{Z}\right)=-\frac{f\ell}{4EA}$, which requires the derivatives of the stiffness matrix $\mathbf{K}$ (and in some cases, the derivative of the load vector $\mathbf{F}$ may not be zero), and solving the finite element equations using the decomposed stiffness matrix $\mathbf{K}$.

As shown in Eq. 18.55, the sensitivity of the stress measure with respect to modulus E is zero, implying that stress is insensitive to the modulus E in this case. In fact, recall Eq. 18.26, the stress of the second element is obtained using FEA:

$$\sigma=Ez_{,1}=E\begin{bmatrix}-\frac{2}{\ell} & \frac{2}{\ell}\end{bmatrix}\cdot\begin{bmatrix}\frac{3f\ell^2}{8EA}\\ \frac{f\ell^2}{2EA}\end{bmatrix}=\frac{f\ell}{4A} \tag{18.56}$$

which does not depend on the modulus E.

For area design variable A, the sensitivity of the stress measure ψ can be calculated using Eq. 18.53:

$$\frac{\mathrm{d}\psi}{\mathrm{d}A}=\frac{\partial\psi}{\partial A}+\frac{\partial\psi}{\partial\mathbf{Z}}\mathbf{K}^{-1}\left(\frac{\partial\mathbf{F}}{\partial \mathrm{A}}-\frac{\partial\mathbf{K}}{\partial \mathrm{A}}\mathbf{Z}\right)$$

$$=0+\begin{bmatrix}\frac{\partial\psi}{\partial Z_2} & \frac{\partial\psi}{\partial Z_3}\end{bmatrix}\left(\frac{\ell}{2EA}\begin{bmatrix}1 & 1\\ 1 & 2\end{bmatrix}\right)\left(\mathbf{0}-\frac{2E}{\ell}\begin{bmatrix}2 & -1\\ -1 & 1\end{bmatrix}\begin{bmatrix}\frac{3f\ell^2}{8EA}\\ \frac{f\ell^2}{2EA}\end{bmatrix}\right) \tag{18.57}$$

$$=0+\begin{bmatrix}-\frac{2E}{\ell} & \frac{2E}{\ell}\end{bmatrix}\begin{bmatrix}-\frac{3f\ell^2}{8EA^2}\\ -\frac{f\ell^2}{2EA^2}\end{bmatrix}=-\frac{f\ell}{4A^2}$$

Equation (18.57) implies that increasing the area reduces stress. Note that, in this case, the explicit dependence term is $\frac{\partial\psi}{\partial A}=0$ because stress is defined as $\sigma=Ez_{,1}$, which does not depend on A explicitly.

18.3.3.2 Adjoint Variable Method

Another way of calculating the implicit dependence $\frac{\partial \mathbf{Z}}{\partial b}$ in Eq. 18.52b is the adjoint variable method. Note that the term $(\partial\psi/\partial\mathbf{Z})\mathbf{K}^{-1}$ in Eq. 18.53 is independent of design. We set the result of $(\partial\psi/\partial\mathbf{Z})\mathbf{K}^{-1}$ as $\boldsymbol{\lambda}^{\mathrm{T}}$:

$$\frac{\partial \psi}{\partial \mathbf{Z}}\mathbf{K}^{-1} = \boldsymbol{\lambda}^{T} \tag{18.58}$$

in which $\boldsymbol{\lambda}$ is called the adjoint response or adjoint solution. Transpose the row vectors in Eq. 18.58 to column vectors:

$$\mathbf{K}^{-\mathrm{T}}\left(\frac{\partial \psi}{\partial \mathbf{Z}}\right)^{\mathrm{T}} = \boldsymbol{\lambda} \tag{18.59}$$

By multiplying both sides of Eq. 18.59 by $\mathbf{K}^{\mathrm{T}}$, we have the following adjoint equation:

$$\mathbf{K}\boldsymbol{\lambda} = \left(\frac{\partial \psi}{\partial \mathbf{Z}}\right)^{\mathrm{T}} \tag{18.60}$$

where the symmetric property of the stiffness matrix, $\mathbf{K} = \mathbf{K}^{\mathrm{T}}$, is used. Note that the adjoint response $\boldsymbol{\lambda}$ of the above adjoint equation can be solved by using the same stiffness matrix $\mathbf{K}$ of the structure. The only difference is the term on the right-hand side, $\partial\psi/\partial\mathbf{Z}$, which is called the adjoint load. The adjoint solution $\boldsymbol{\lambda}$ is not dependent on design but on the performance measure ψ. Thus, the adjoint solution is required per performance measure. Once the adjoint solution is available, the sensitivity can be calculated from Eq. 18.53 as

$$\frac{\mathrm{d}\psi}{\mathrm{d}b} = \frac{\partial \psi}{\partial b} + \boldsymbol{\lambda}^{\mathrm{T}}\left(\frac{\partial \mathbf{F}}{\partial b} - \frac{\partial \mathbf{K}}{\partial b}\mathbf{Z}\right) \tag{18.61}$$

Calculating the sensitivity using Eq. 18.61 is called the adjoint variable method—or more specifically, the adjoint variable method of the discrete-analytical approach.

We consider a general design problem of linear static with number of design variables (NDV) and number of performance measures (NPF) defined in the number of load cases (NL). Note that for a static problem of multiple load cases, performance measures such as stresses or displacements can be defined for respective load cases. If we use the direct differentiation method shown in Eq. 18.53, we need to calculate $\partial\mathbf{Z}/\partial b_i$, $i = 1$, NDV for NL $\times$ NDV times by solving Eq. 18.41. If we use the adjoint variable method, we need to solve Eq. 18.60 NPF times. Solving the matrix equations for additional fictitious loads or adjoint loads constitutes the major computation time in calculating sensitivity coefficients. Therefore, when the number of performance measures is greater than that of the number of design variables times the number of load cases—that is, NPF $>$ NL $\times$ NDV—the direct differential method is more efficient. Otherwise, the adjoint variable method is more desirable computation-wise.

EXAMPLE 18.5

Recall the displacement performance measure defined at the bottom end of the bar shown in Figure 18.4, written as

$$\psi = z(\ell) = \frac{f\ell^2}{2EA} = Z_3 \tag{18.62}$$

Calculate the adjoint load for the displacement performance measure, and calculate its sensitivity for the modulus design variable E using the adjoint variable method.

Solution

In finite element formulation, the displacement of the bar is written as in Eq. 18.24. Because the performance measure is the displacement at the bottom end, which involves only the second finite element, the adjoint load of the displacement measure ψ can be obtained using Eq. 18.60:

$$\frac{\partial \psi}{\partial \mathbf{Z}} = \begin{bmatrix} \frac{\partial \psi}{\partial Z_2} & \frac{\partial \psi}{\partial Z_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial Z_3}{\partial Z_2} & \frac{\partial Z_3}{\partial Z_3} \end{bmatrix} = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{18.63}$$

Hence, the adjoint load for the displacement is nothing but a unit force acting downward (in the same direction as the displacement measure) at node 3 (the location where the displacement measure is defined).

The adjoint responses can be calculated as

$$\boldsymbol{\lambda} = \mathbf{K}^{-1}\left(\frac{\partial \psi}{\partial \mathbf{Z}}\right)^{\mathrm{T}} = \left(\frac{\ell}{2EA}\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}\right)\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\ell}{2EA} \\ \frac{\ell}{EA} \end{bmatrix} \tag{18.64}$$

Hence, the sensitivity of the stress measure can be calculated using Eq. 18.61:

$$\begin{aligned} \frac{\mathrm{d}\psi}{\mathrm{d}E} &= \frac{\partial \psi}{\partial E} + \boldsymbol{\lambda}^{\mathrm{T}}\left(\frac{\partial \mathbf{F}}{\partial E} - \frac{\partial \mathbf{K}}{\partial E}\mathbf{Z}\right) \\ &= 0 + \begin{bmatrix} \frac{\ell}{2EA} & \frac{\ell}{EA} \end{bmatrix}\left(-\frac{2A}{\ell}\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} \frac{3f\ell^2}{8EA} \\ \frac{f\ell^2}{2EA} \end{bmatrix}\right) = -\frac{f\ell^2}{2E^2A} \end{aligned} \tag{18.65}$$

This can also be obtained by taking derivative of ψ ($\psi = Z_3$) with respect to E directly using Eq. 18.62.

EXAMPLE 18.6

Repeat Example 18.4 using the adjoint variable method, only for modulus design variable E.

Solution

The stress of the second element of the simple bar is obtained in Example 18.4 as

$$\psi = \sigma = E\begin{bmatrix} -\frac{2}{\ell} & \frac{2}{\ell} \end{bmatrix} \cdot \begin{bmatrix} Z_2 \\ Z_3 \end{bmatrix} \tag{18.66}$$

Hence, the adjoint load of the stress measure can be obtained using Eq. 18.60:

$$\frac{\partial \psi}{\partial \mathbf{Z}} = \begin{bmatrix} \frac{\partial \psi}{\partial Z_2} & \frac{\partial \psi}{\partial Z_3} \end{bmatrix} = \begin{bmatrix} -\frac{2E}{\ell} & \frac{2E}{\ell} \end{bmatrix} \tag{18.67}$$

Continued

EXAMPLE 18.6—cont'd

This consists of a point force $-2E/\ell$ (acting in the upward direction) at node 2, and another point force $2E/\ell$ (acting in the downward direction at node 3). The adjoint responses are

$$\boldsymbol{\lambda} = \mathbf{K}^{-1}\left(\frac{\partial \psi}{\partial \mathbf{Z}}\right)^{\mathrm{T}} = \left(\frac{\ell}{2EA}\begin{bmatrix}1 & 1\\ 1 & 2\end{bmatrix}\right)\begin{bmatrix}-\dfrac{2E}{\ell}\\ \dfrac{2E}{\ell}\end{bmatrix} = \begin{bmatrix}0\\ \dfrac{1}{A}\end{bmatrix} \tag{18.68}$$

Hence, the sensitivity of the stress measure can be calculated using Eq. 18.61, for example, assuming modulus E as the design variable:

$$\begin{aligned}\frac{\mathrm{d}\psi}{\mathrm{d}E} &= \frac{\partial \psi}{\partial E} + \boldsymbol{\lambda}^{\mathrm{T}}\left(\frac{\partial \mathbf{F}}{\partial E} - \frac{\partial \mathbf{K}}{\partial E}\mathbf{Z}\right)\\ &= \begin{bmatrix}-\dfrac{2}{\ell} & \dfrac{2}{\ell}\end{bmatrix}\cdot\begin{bmatrix}\dfrac{3f\ell^2}{8EA}\\ \dfrac{f\ell^2}{2EA}\end{bmatrix} + \begin{bmatrix}0 & \dfrac{1}{A}\end{bmatrix}\left(-\frac{2A}{\ell}\begin{bmatrix}2 & -1\\ -1 & 1\end{bmatrix}\begin{bmatrix}\dfrac{3f\ell^2}{8EA}\\ \dfrac{f\ell^2}{2EA}\end{bmatrix}\right) = 0\end{aligned} \tag{18.69}$$

18.3.4 CONTINUUM APPROACH

In the continuum approach, the sensitivity is obtained by taking the variation of the energy equation that governs the structural behavior. The energy equation of a special case—the simple bar example—can be found in Eq. 18.13:

$$a_b(z, \overline{z}) = \int_0^{\ell} EAz_{,1}\overline{z}_{,1}\mathrm{d}x = \int_0^{\ell} f\,\overline{z}\mathrm{d}x = \ell_b(\overline{z}), \quad \forall \overline{z} \in S \tag{18.70}$$

in which the subscript b emphasizes the dependence of energy forms on design b.

A variation on the energy equation leads to a set of sensitivity equations in integral form that are then solved numerically, usually (but not necessarily) with the same discretization in a finite element formulation as was used for the original structural response. This is the continuum-discrete method. For some simple but rare cases, state variable $\mathbf{z}$ can be obtained analytically, which can then be plugged into the sensitivity equations to calculate the sensitivity coefficients in integral form without discretization. This is called the continuum-analytical method. The continuum-discrete method is general and will be the focus of our discussion.

Before formally introducing the variation operator, we simply state that the variation of a function $f(b)$ is written as

$$f' = \frac{\partial f}{\partial b}\delta b = f'_{\delta b} \tag{18.71}$$

in which δb is the variation in the variable b. More about variation will be discussed in Section 18.4.2.

18.3.4.1 Direction Differentiation Method

We take variation of the energy bilinear form of the bar example, considering, for example, the area A as the design variable:

$$\left[a_b(z,\overline{z})\right]' = \left[\int_0^\ell EAz_{,1}\overline{z}_{,1}\mathrm{d}x\right]' = \int_0^\ell Ez_{,1}\overline{z}_{,1}\mathrm{d}x\delta A + \int_0^\ell EAz'_{,1}\overline{z}_{,1}\mathrm{d}x = a'_{\delta b}(z,\overline{z}) + a_b(z',\overline{z}) \tag{18.72}$$

where $a'_{\delta b}(z,\overline{z}) = \int_0^\ell E\delta Az_{,1}\overline{z}_{,1}\mathrm{d}x$ represents the explicit dependence of the energy bilinear form on design, and $a_b(z',\overline{z}) = \int_0^\ell EAz'_{,1}\overline{z}_{,1}\mathrm{d}x$ shows the implicit dependence through $z'_{,1}$. According to Eq. 18.71, $z'_{,1} = \frac{\partial z_{,1}}{\partial A}\delta A$. Now, the variation of the load linear form is written as

$$\left[\ell_b(\overline{z})\right]' = \ell'_{\delta b}(\overline{z}) \tag{18.73}$$

Hence, the sensitivity equation of the continuum approach can be obtained as

$$a_b(z',\overline{z}) = -a'_{\delta b}(z,\overline{z}) + \ell'_{\delta b}(\overline{z}), \quad \forall \overline{z} \in S \tag{18.74}$$

The term on the left-hand side of Eq. 18.74 is similar to that of Eq. 18.70, except that displacement z in Eq. 18.70 is replaced by z′ in Eq. 18.74. The right-hand side of Eq. 18.74 defines a fictitious load, which is written explicitly in terms of the design variable. Thus, solving the sensitivity equation (Eq. 18.74) is the same as solving the original structural equilibrium equation (Eq. 18.70) with fictitious load(s).

For the simple bar example, $[\ell_b(\overline{z})]' = [\int_0^\ell f\,\overline{z}\mathrm{d}x]' = \int_0^\ell f'\,\overline{z}\mathrm{d}x\delta A = 0 = \ell'_{\delta b}(\overline{z})$ because f is the distributed load inside the bar, which was assumed to be constant; that is, $f' = 0$. Hence, the sensitivity equation for the bar example can be written as

$$\int_0^\ell EAz'_{,1}\overline{z}_{,1}\mathrm{d}x = -\int_0^\ell E\delta Az_{,1}\overline{z}_{,1}\mathrm{d}x \tag{18.75}$$

EXAMPLE 18.7

Calculate the sensitivity of the displacement at the bottom end of the simple bar example with respect to area design variable A using the direct differential method of the continuum-discrete approach.

Solution

We discretize the sensitivity equation of Eq. 18.75 using the same shape function defined in Eq. 18.17:

$$z(x) = \begin{cases} \left[1 - \dfrac{2x}{\ell} \quad \dfrac{2x}{\ell}\right] \cdot \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}, & 0 \le x \le \ell/2 \\ \left[1 - \dfrac{2(x-\ell/2)}{\ell} \quad \dfrac{2(x-\ell/2)}{\ell}\right] \cdot \begin{bmatrix} Z_2 \\ Z_3 \end{bmatrix}, & \ell/2 \le x \le \ell \end{cases} \tag{18.76}$$

Continued

EXAMPLE 18.7—cont'd

We have the finite element matrix equations on the left-hand side (similar to Eq. 18.18), as follows:

$$\int_0^{\ell} EAz'_{,1}\overline{z}_{,1}\mathrm{d}x \overset{\text{discretize}}{=} \begin{bmatrix}\overline{Z}_1\\ \overline{Z}_2\\ \overline{Z}_3\end{bmatrix}^{\mathrm{T}} \cdot \frac{2EA}{\ell}\begin{bmatrix}1 & -1 & 0\\ -1 & 2 & -1\\ 0 & -1 & 1\end{bmatrix} \cdot \begin{bmatrix}Z'_1\\ Z'_2\\ Z'_3\end{bmatrix} = \overline{\mathbf{Z}}_{\mathrm{g}}\mathbf{K}_{\mathrm{g}}\mathbf{Z}'_{\mathrm{g}} \tag{18.77}$$

where

$$\mathbf{Z}'_{\mathrm{g}} = \begin{bmatrix}Z'_1\\ Z'_2\\ Z'_3\end{bmatrix} = \begin{bmatrix}\dfrac{\partial Z_1}{\partial A}\\ \dfrac{\partial Z_2}{\partial A}\\ \dfrac{\partial Z_3}{\partial A}\end{bmatrix}\delta A \tag{18.78}$$

Then, the right-hand side of Eq. 18.75 is discretized with the same shape function:

$$\begin{aligned}
&-\int_0^{\ell} E z_{,1}\overline{z}_{,1}\mathrm{d}x\delta A = -\int_0^{\ell/2} E z_{,1}\overline{z}_{,1}\mathrm{d}x\delta A - \int_{\ell/2}^{\ell} E z_{,1}\overline{z}_{,1}\mathrm{d}x\delta A\\
&= -\int_0^{\ell/2} E\left\{\begin{bmatrix}\overline{Z}_1 & \overline{Z}_2\end{bmatrix}\begin{bmatrix}-\dfrac{2}{\ell}\\ \dfrac{2}{\ell}\end{bmatrix}\right\}\frac{3f\ell}{4EA}\,\mathrm{d}x\delta A - \int_{\ell/2}^{\ell} E\left\{\begin{bmatrix}\overline{Z}_2 & \overline{Z}_3\end{bmatrix}\begin{bmatrix}-\dfrac{2}{\ell}\\ \dfrac{2}{\ell}\end{bmatrix}\right\}\frac{f\ell}{4EA}\,\mathrm{d}x\delta A\\
&= -\begin{bmatrix}\overline{Z}_1 & \overline{Z}_2\end{bmatrix}\begin{bmatrix}-\dfrac{3f\ell}{4A}\\ \dfrac{3f\ell}{4A}\end{bmatrix}\delta A - \begin{bmatrix}\overline{Z}_2 & \overline{Z}_3\end{bmatrix}\begin{bmatrix}-\dfrac{f\ell}{4A}\\ \dfrac{f\ell}{4A}\end{bmatrix}\delta A = -\begin{bmatrix}\overline{Z}_1 & \overline{Z}_2 & \overline{Z}_3\end{bmatrix}\begin{bmatrix}\dfrac{3f\ell}{4A}\\ -\dfrac{f\ell}{2A}\\ -\dfrac{f\ell}{4A}\end{bmatrix}\delta A = \overline{\mathbf{Z}}_{\mathrm{g}}\mathbf{F}_{\mathrm{g}}\delta A
\end{aligned} \tag{18.79}$$

By equating Eq. 18.79 with 18.77 and applying boundary condition, we have $\mathbf{KZ}' = \mathbf{F}_{\mathrm{fic}}\delta A$:

$$\mathbf{KZ}' = \frac{2EA}{\ell}\begin{bmatrix}2 & -1\\ -1 & 1\end{bmatrix}\cdot\begin{bmatrix}\dfrac{\partial Z_2}{\partial A}\\ \dfrac{\partial Z_3}{\partial A}\end{bmatrix}\delta A = \begin{bmatrix}-\dfrac{f\ell}{2A}\\ -\dfrac{f\ell}{4A}\end{bmatrix}\delta A = \mathbf{F}_{\mathrm{fic}}\delta A \tag{18.80}$$

Solving Eq. 18.80 and removing δA, we have

$$\begin{bmatrix}\dfrac{\partial Z_2}{\partial A}\\ \dfrac{\partial Z_3}{\partial A}\end{bmatrix} = \mathbf{K}^{-1}\mathbf{F}_{\mathrm{fic}} = \frac{\ell}{2EA}\begin{bmatrix}1 & 1\\ 1 & 2\end{bmatrix}\begin{bmatrix}-\dfrac{f\ell}{2A}\\ -\dfrac{f\ell}{4A}\end{bmatrix} = \begin{bmatrix}-\dfrac{3f\ell^2}{8EA^2}\\ -\dfrac{f\ell^2}{2EA^2}\end{bmatrix} \tag{18.81}$$

The major advantage of the continuum approach is that the sensitivity formulation is independent of the discrete model and numerical schemes. Once the continuum sensitivity equation is obtained, it can be discretized in the same manner as the original analysis. In addition, there is no need to take partial derivatives of the matrix or vector, as required, in the discrete approach, and no need to find design perturbation sizes, as is needed in the semi-analytical and overall finite difference methods. More importantly, for practical implementation, the continuum-discrete method can be implemented outside of commercial FEA codes by only using the FEA results. More details are provided in Section 18.4.4.

18.3.4.2 Adjoint Variable Method

In the continuum approach, the performance measure is expressed in an integral form. For example, the displacement at the bottom end of the bar is defined as

$$\psi = z(\ell) = \int_0^\ell \left[z\left(x\right) \widehat{\delta}\left(x-\ell\right) \right] \mathrm{d}x \tag{18.82}$$

Here, $\widehat{\delta}(x-\ell)$ is a direct delta function, defined as

$$\widehat{\delta}(x-\ell) \equiv \begin{cases} \infty, & x=\ell \\ 0, & x \neq \ell \end{cases} \tag{18.83}$$

Taking the variation of Eq. 18.82, we have

$$\psi' = \int_0^\ell \left[z'\left(x\right) \widehat{\delta}\left(x-\ell\right) \right] \mathrm{d}x \tag{18.84}$$

The basic idea of the adjoint variable method is to avoid directly calculating z' in Eq. 18.84 by introducing an adjoint structure that is identical to the original structure but with a different set of load—that is, the adjoint load. The adjoint structure for the simple bar example is defined as

$$a_b\left(\lambda, \overline{\lambda}\right) = \int_0^\ell \left[\widehat{\delta}\left(x-\ell\right) \overline{\lambda}(x) \right] \mathrm{d}x, \quad \forall\, \overline{\lambda} \in S \tag{18.85}$$

whose right-hand side is identical to that of Eq. 18.84 by replacing z' with the virtual adjoint response $\overline{\lambda}$. Note that the left-hand side of Eq. 18.85 is the energy bilinear form in terms of adjoint response λ and virtual adjoint response $\overline{\lambda}$. The right-hand side defines the adjoint load, which depends on the type of performance measure. The adjoint response can be obtained in the same way as the original structural response z, for example, using finite element equations, as long as the adjoint load vector is available.

We first evaluate Eq. 18.85 at $\overline{\lambda} = z'$:

$$a_b(\lambda, z') = \int_0^\ell \left[\widehat{\delta}\left(x-\ell\right) z'\left(x\right) \right] \mathrm{d}x, \quad \forall\, z' \in S \tag{18.86}$$

in which the right-hand side equals that of Eq. 18.84.

Recall the sensitivity equation of the direct differentiation method in Eq. 18.74:

$$a_b(z', \bar{z}) = -a'_{\delta b}(z, \bar{z}) + \ell'_{\delta b}(\bar{z}), \quad \forall \bar{z} \in S \tag{18.87}$$

We evaluate Eq. 18.74 at $\bar{z} = \lambda$:

$$a_b(z', \lambda) = -a'_{\delta b}(z, \lambda) + \ell'_{\delta b}(\lambda) \tag{18.88}$$

Because the energy bilinear form is symmetric—that is, $a_b(\lambda, z') = a_b(z', \lambda)$—Eq. 18.88 equals Eq. 18.86. Therefore,

$$\psi' = \int_0^\ell \left[\hat{\delta}(x - \ell)z'(x)\right] dx = -a'_{\delta b}(z, \lambda) + \ell'_{\delta b}(\lambda) \tag{18.89}$$

This can be calculated once the original structural response z and adjoint response λ are available. Note that additional term(s) may exist in Eq. 18.89 for performance measures other than displacement, which will be illustrated in Section 18.4.3.

EXAMPLE 18.8

Solve for the same problem as Example 18.7 using the adjoint variable method. We defined displacement at node 3 as the performance measure, which is written in an integral form as

$$\psi = z(\ell) = \int_0^\ell \left[z(x)\hat{\delta}(x - \ell)\right] dx \tag{18.90}$$

Solution

We first create an adjoint structure by determining the adjoint load applied to the simple bar. From Eq. 18.85, we have

$$a_b(\lambda, \bar{\lambda}) = \int_0^\ell \left[\hat{\delta}(x - \ell)\bar{\lambda}(x)\right] dx \tag{18.91}$$

Discretizing Eq. 18.91 using finite element shape function, we have

$$\begin{aligned} a_b(\lambda, \bar{\lambda}) &= \int_0^\ell \left[\hat{\delta}(x - \ell)\bar{\lambda}(x)\right] dx = \int_0^{\ell/2} \left[\hat{\delta}(x - \ell)\bar{\lambda}(x)\right] dx + \int_{\ell/2}^\ell \left[\hat{\delta}(x - \ell)\bar{\lambda}(x)\right] dx \\ &= 0 + \int_{\ell/2}^\ell \left[\hat{\delta}(x - \ell)\bar{\lambda}(x)\right] dx = \begin{bmatrix} \bar{\lambda}_2 & \bar{\lambda}_3 \end{bmatrix} \int_{\ell/2}^\ell \left[\hat{\delta}(x - \ell) \begin{bmatrix} 1 - \dfrac{2(x - \ell/2)}{\ell} \\ \dfrac{2(x - \ell/2)}{\ell} \end{bmatrix}\right] dx \\ &= \begin{bmatrix} \bar{\lambda}_2 & \bar{\lambda}_3 \end{bmatrix} \begin{bmatrix} 1 - \dfrac{2(\ell - \ell/2)}{\ell} \\ \dfrac{2(\ell - \ell/2)}{\ell} \end{bmatrix} = \begin{bmatrix} \bar{\lambda}_2 & \bar{\lambda}_3 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{aligned} \tag{18.92}$$

which shows that the adjoint load is a point load applied at the bottom end of the bar (at node 3) in the downward direction, which is identical to those obtained using the discrete approach.

EXAMPLE 18.8—cont'd

Solve the adjoint response by using the finite element equation:

$$\boldsymbol{\lambda} = \mathbf{K}^{-1}\mathbf{F} = \frac{\ell}{2EA}\begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{\ell}{2EA} \\ \dfrac{\ell}{EA} \end{bmatrix} \tag{18.93}$$

Bring the original responses z and adjoint responses λ to the sensitivity equation (Eq. 18.89):

$$\begin{aligned} \psi' &= -a'_{\delta b}(z,\lambda) + \ell'_{\delta b}(\lambda) = -\int_0^{\ell} E z_{,1}\lambda_{,1}\,\mathrm{d}x\delta A + 0 \\ &= -\int_0^{\ell/2} E\frac{3f\ell}{4EA}\begin{bmatrix} -\dfrac{2}{\ell} & \dfrac{2}{\ell} \end{bmatrix}\cdot\begin{bmatrix} 0 \\ \dfrac{\ell}{2EA} \end{bmatrix}\mathrm{d}x\delta A - \int_{\ell/2}^{\ell} E\frac{f\ell}{4EA}\begin{bmatrix} -\dfrac{2}{\ell} & \dfrac{2}{\ell} \end{bmatrix}\cdot\begin{bmatrix} \dfrac{\ell}{2EA} \\ \dfrac{\ell}{EA} \end{bmatrix}\mathrm{d}x\delta A \\ &= -\frac{f\ell^2}{2E^2A} \end{aligned} \tag{18.94}$$

18.4 SIZING AND MATERIAL DESIGNS

We briefly introduced three general approaches: finite difference, discrete, and continuum, for calculating the sensitivity of structural performance measures. In these approaches, the overall finite difference is the simplest and easiest to implement with commercial finite element codes. However, the finite difference is inefficient and requires adequate design perturbations. The gradient-based optimization method implemented in Pro/MECHANICA Structure (www.ptc.com) uses the overall finite difference approach for sensitivity analysis. The discrete approach formulates the sensitivity expressions by taking the derivatives of the discretized finite element equations. The derivatives of the finite element equations can be obtained analytically if the explicit expressions of the equations in terms of design variables are available, which is called the discrete-analytical approach. If not, the finite difference method is employed to approximate the derivatives of the equations. This approach is called the discrete–discrete or semi-analytical method. The solution 200: Design Optimization and Sensitivity Analysis offered by MSC/NASTRAN (www.mscsoftware.com) employs the semi-analytical approach for sensitivity analysis. The discrete approach is more efficient than the overall finite difference. However, the semi-analytical method still requires adequate design perturbations for accurate sensitivity information. The continuum approach formulates the sensitivity expressions by taking a variation of the energy equations of the structure in an integral form. The expressions can then be discretized using finite element shape functions for numerical evaluations, which is called the continuum-discrete approach. This approach is efficient and does not require any design perturbations.

Considering accuracy and efficiency, the best method is probably the continuum-discrete. In this section, we formally introduce this method. We include sensitivity analysis for sizing and material

design variables that do not affect the geometric shape of the structures; therefore, the sensitivity analysis methods and formulations are similar. In Section 18.5, we introduce shape sensitivity analysis, in which design variables alter the geometric shape of the structure. The shape sensitivity analysis method and formulations are quite different from those of the sizing and material design variables.

We start by introducing energy equations for a general structure in Section 18.4.1. We formally introduce variation in Section 18.4.2, and then focus on static problems in Section 18.4.3, in which we use beam examples for illustration. We discuss implementation of the continuum-discrete approach with commercial FEA codes in Section 18.4.4.

18.4.1 PRINCIPLE OF VIRTUAL WORK

We discussed the energy equation and principle of virtual work for the simple bar example in Section 18.3. The energy equation or the principle of virtual work is the basis of FEA and sensitivity analysis, either the discrete or continuum approach. In this subsection, we present the energy equation of a general two-dimensional structure. Equations for three-dimensional structures can also be obtained with more complex formulations. Similar to the simple bar example, we start by stating the governing equation in differential form, and then carrying out integration by parts to obtain the energy equations. Note that for most structural problems, such as the simple bar example, energy equations can be obtained by starting with the differential equations and carrying out integration by parts. The energy equations for a specific problem and type of structure can also be obtained by using the energy equation of the general structure, with terms and state variables specialized to the specific problem and type of the structure at hand. We use a beam example to illustrate this point in Section 18.4.3.

18.4.1.1 Differential Equation

A general two-dimensional or planar structure shown in Figure 18.7(a) is loaded with a traction force (force per length) $\mathbf{T} = [T_1, T_2]^{\mathrm{T}}$ at the boundary Γ_1 and an in-plane distributed force (or body force) per area $\boldsymbol{f} = [f_1, f_2]^{\mathrm{T}}$, and fixed at the boundary Γ_0. Note that T_1 and T_2 are the traction forces in the x_1 and x_2 directions, respectively. The same is true for f_1 and f_2. The stress element A inside the structure domain Ω is shown in Figure 18.7(b), in which three stress components σ_{11}, σ_{22}, and σ_{12} are present. Note that σ_{12} is the shear stress τ_{12}, and $\sigma_{12} = \sigma_{21}$. A stress element B at the boundary Γ_1 is

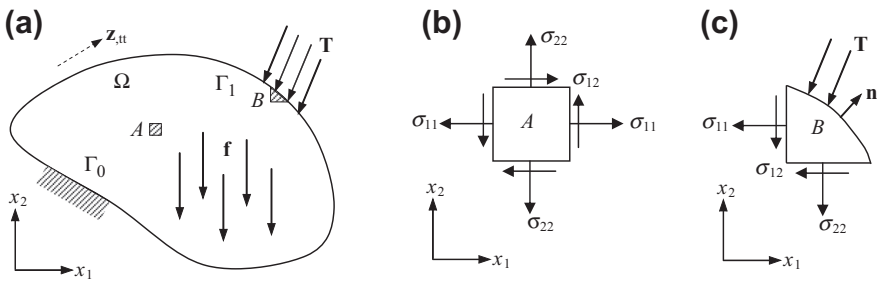


FIGURE 18.7

A general structure with (a) load and boundary conditions, (b) an internal stress element, and (c) a stress element at the traction boundary Γ_1.

shown in Figure 18.7(c), in which $\mathbf{n} = [n_1, n_2]^{\mathrm{T}}$ is the normal vector of a given point at boundary Γ_1. For static problems, forces $\boldsymbol{f}$ and $\boldsymbol{T}$ are independent of time; hence, the displacement solution $\mathbf{z} = [z_1, z_2]^{\mathrm{T}}$ is not a function of time. For dynamic problems, either $\mathbf{f}$ or $\mathbf{T}$ or both forces are time functions, in which the structure is involved with inertia force caused by acceleration $\mathbf{z}_{,\mathrm{tt}}$. Note that the shorthand notation, tt in subscript denotes the second derivative of the function with respect to time t. The displacement $\mathbf{z}$ is function of spatial variable $\mathbf{x}$ and time t—that is, $\mathbf{z} = \mathbf{z}(\mathbf{x}, t)$, in this case.

The governing equations in differential form are stated as follows:

$$\begin{cases} \sigma_{11,1} + \sigma_{12,2} + f_1 = \rho z_{1,tt} \\ \sigma_{12,1} + \sigma_{22,2} + f_2 = \rho z_{2,tt} \end{cases}, \quad \text{in } \Omega \tag{18.95a}$$

The boundary conditions are

$$T_1 = \sigma_{11} n_1 + \sigma_{12} n_2 \quad \text{and} \quad T_2 = \sigma_{21} n_1 + \sigma_{22} n_2, \quad \text{on} \quad \Gamma_1, \quad \text{and} \tag{18.95b}$$

$$z_1(x,t) = 0 \quad \text{and} \quad z_2(x,t) = 0, \quad \text{on} \quad \Gamma_0 \tag{18.95c}$$

The initial displacement and initial velocity conditions, respectively, are

$$z_1(\mathbf{x}, 0) = z_1^0 \quad \text{and} \quad z_2(\mathbf{x}, 0) = z_2^0, \quad \text{in} \quad \Omega \tag{18.95d}$$

$$z_{1,t}(\mathbf{x}, 0) = z_{1,t}^0 \quad \text{and} \quad z_{2,t}(\mathbf{x}, 0) = z_{2,t}^0, \quad \text{in} \quad \Omega \tag{18.95e}$$

Note that in Eq. 18.95a, the shorthand notations of subscript 1 and 2 represent the derivatives of the function with respect to x_1 and x_2, respectively.

Equations (18.96a) to (18.96e) can be further written in a compact form as

$$\sigma_{\mathrm{ij,j}} + f_i = \rho z_{i,tt}, \quad \text{in} \quad \Omega \tag{18.96a}$$

The boundary and initial conditions are

$$T_i = \sigma_{\mathrm{ij}} n_j, \quad \text{on} \quad \Gamma_{1,} \quad \text{and} \tag{18.96b}$$

$$z_i(\mathbf{x}, t) = 0, \quad \text{on} \quad \Gamma_0 \tag{18.96c}$$

$$z_i(\mathbf{x}, 0) = z_i^0, \quad \text{in} \quad \Omega \tag{18.96d}$$

$$z_{i,t}(\mathbf{x}, 0) = z_{i,t}^0, \quad \text{in} \quad \Omega \tag{18.96e}$$

For two-dimensional problems, $i, j = 1, 2$; for three-dimensional problems, $i, j = 1, 3$.

If we assume that the external loads do not depend on time t, the differential form of the governing equations of the structure shown in Figure 18.7(a) is reduced to

$$\sigma_{\mathrm{ij,j}} + f_i = 0, \quad \text{in} \quad \Omega \tag{18.97a}$$

with boundary conditions

$$T_i = \sigma_{\mathrm{ij}} n_j, \quad \text{on} \quad \Gamma_{1,} \quad \text{and} \tag{18.97b}$$

$$z_i(x) = 0, \quad \text{on} \quad \Gamma_0 \tag{18.97c}$$

18.4.1.2 Energy Formulation

Similar to the simple bar example, we derive the energy equation of the structural governing equation by multiplying both sides of Eq. 18.97a with an arbitrary virtual displacement $\overline{\mathbf{z}}$ and then integrating over the structural domain Ω:

$$\int_{\Omega} \sigma_{\mathrm{ij,j}} \overline{z}_i \mathrm{d}\Omega + \int_{\Omega} f_i \overline{z}_i \mathrm{d}\Omega = 0 \tag{18.98}$$

Carrying out integration by part for the terms on the left-hand side of Eq. 18.98, we have

$$\int_{\Gamma_1} \sigma_{\mathrm{ij}} \overline{z}_i n_j \mathrm{d}\Gamma_1 - \int_{\Omega} \sigma_{\mathrm{ij}} \overline{z}_{\mathrm{i,j}} \mathrm{d}\Omega + \int_{\Omega} f_i \overline{z}_i \mathrm{d}\Omega = 0 \tag{18.99}$$

in which the first term on the left-hand side can be evaluated, using the boundary condition of Eq. 18.97b, as

$$\int_{\Gamma_1} \sigma_{\mathrm{ij}} \overline{z}_i n_j \mathrm{d}\Gamma_1 = \int_{\Gamma_1} T_i \overline{z}_i \mathrm{d}\Gamma_1 \tag{18.100}$$

The second term on the left-hand side can be written as

$$\int_{\Omega} \sigma_{\mathrm{ij}} \overline{z}_{\mathrm{i,j}} \mathrm{d}\Omega = \int_{\Omega} \sigma_{\mathrm{ij}} \overline{\varepsilon}_{\mathrm{ij}} \mathrm{d}\Omega \tag{18.101}$$

in which $\overline{\varepsilon}_{\mathrm{ij}}$ is the virtual strain in virtual displacement $\overline{z}_i$. Note that the relationship $\sigma_{\mathrm{ij}} \overline{z}_{\mathrm{i,j}} = \sigma_{\mathrm{ij}} \overline{\varepsilon}_{\mathrm{ij}}$ is employed in Eq. 18.101, for the following two reasons:

$$\sigma_{\mathrm{ij}} \overline{z}_{\mathrm{i,j}} = \sigma_{\mathrm{ji}} \overline{z}_{\mathrm{j,i}} = \sigma_{\mathrm{ij}} \overline{z}_{\mathrm{j,i}} \tag{18.102}$$

and

$$\sigma_{\mathrm{ij}} \overline{z}_{\mathrm{i,j}} = \sigma_{\mathrm{ji}} \left[\frac{1}{2} \left(\overline{z}_{\mathrm{i,j}} + \overline{z}_{\mathrm{j,i}} \right) \right] = \sigma_{\mathrm{ij}} \overline{\varepsilon}_{\mathrm{ij}} \tag{18.103}$$

Note that in Eq. 18.102, $\sigma_{\mathrm{ij}} \overline{z}_{\mathrm{i,j}} = \sigma_{\mathrm{ji}} \overline{z}_{\mathrm{j,i}}$ is simply obtained by changing the index, and $\sigma_{\mathrm{ji}} \overline{z}_{\mathrm{j,i}} = \sigma_{\mathrm{ij}} \overline{z}_{\mathrm{j,i}}$ is due to the symmetric of the stress tensor σ_{ij} (e.g., $\sigma_{12} = \sigma_{21}$). In Eq. 18.103, the following relationship, called the Cauchy strain tensor, is employed:

$$\overline{\varepsilon}_{\mathrm{ij}} = \frac{1}{2} \left(z_{\mathrm{i,j}} + z_{\mathrm{j,i}} \right) \tag{18.104}$$

Bringing Eqs 18.100 and 18.101 into Eq. 18.99, we have

$$\int_{\Omega} \sigma_{\mathrm{ij}} \overline{\varepsilon}_{\mathrm{ij}} \mathrm{d}\Omega = \int_{\Omega} f_i \overline{z}_i \mathrm{d}\Omega + \int_{\Gamma_1} T_i \overline{z}_i \mathrm{d}\Gamma_1 \tag{18.105}$$

Equation (18.105) can be rewritten as

$$a(\mathbf{z}, \overline{\mathbf{z}}) = \ell(\overline{\mathbf{z}}) \tag{18.106}$$

Here, $a(\mathbf{z}, \overline{\mathbf{z}})$ and $\ell(\overline{\mathbf{z}})$ are the energy bilinear form (or virtual strain energy) and load linear form (virtual work), defined respectively as

$$a(\mathbf{z}, \overline{\mathbf{z}}) = \int_{\Omega} \sigma_{ij} \overline{\varepsilon}_{ij} d\Omega \tag{18.107}$$

and

$$\ell(\overline{\mathbf{z}}) = \int_{\Omega} f_i \overline{z}_i d\Omega + \int_{\Gamma_1} T_i \overline{z}_i d\Gamma_1 \tag{18.108}$$

The principle of virtual work states, from a structural analysis perspective, that the solution $\mathbf{z}$ is sought in a trial function space S. That is, $\mathbf{z} \in S$, which is the space of kinematically admissible virtual displacement, defined as

$$S = \left\{ \mathbf{z} \in [H^m(\Omega)]^2 \middle| \mathbf{z} = 0 \quad \text{at} \ \Gamma_0 \right\} \tag{18.109}$$

such that Eq. 18.106 is satisfied for all virtual displacements $\overline{\mathbf{z}} \in S$.

In Eq. 18.109, $\mathbf{z} = 0$ at Γ_0 is the essential (or kinematic) boundary condition and $[H^m(\Omega)]^2$ is the Sobolev space of order m, where m is a positive integer, determined by the type of the structural components—for example $m = 1$ for bars and $m = 2$ for beams. The superscript 2 implies a two-dimensional problem. Similar to the discussion in Section 18.2 for the bar example, the solution $\mathbf{z}$ in $[H^m(\Omega)]^2$ implies that the virtual energy terms $a(\mathbf{z}, \overline{\mathbf{z}})$ and $\ell(\overline{\mathbf{z}})$ in Eq. 18.106 are finite. Without going through rigorous mathematical arguments, we state that the solutions $\mathbf{z}$ of Eq. 18.106 exist and are unique.

18.4.2 VARIATIONS

In Section 18.3.4, we simply stated that the variation of a function $f(b)$ is written as

$$f' = \frac{\partial f}{\partial b} \delta b = f'_{\delta b} \tag{18.110}$$

In fact, the variation we stated is the first variation in the Calculus of Variations.

18.4.2.1 Definition

In the continuum approach, sensitivity can be understood as variation of a function. Let us consider that a vector of design variable $\mathbf{b}$ is perturbed to $\mathbf{b} + \tau\delta\mathbf{b}$, in which τ is a scalar that measures the perturbation size and $\delta\mathbf{b}$ is the direction of design change. For simplicity, we assume for the time being material or sizing design variables that are not affecting the geometric shape of the structure. Shape sensitivity analysis will be discussed in Section 18.5. We also assume a single design variable b for the time being. The variation of the state variable (assuming a scalar for simplicity), $z(x, b)$, which is a function of both spatial variable x (location) and design variable b, is defined as

$$z' \equiv \frac{d}{d\tau} z(x, b + \tau\delta b) \bigg|_{\tau=0} = \lim_{\tau \to 0} \frac{z(x, b + \tau\delta b) - z(x, b)}{\tau} \tag{18.111}$$

Taking Taylor's series expansion for $z(x, b + \tau\delta b)$ at b, we have

$$z(x, b + \tau\delta b) = z(x, b) + \frac{\partial z}{\partial b}\tau\delta b + \frac{1}{2}\frac{\partial^2 z}{\partial b^2}(\tau\delta b)^2 + \text{ higher order terms} \tag{18.112}$$

Plugging Eq. 18.112 into Eq. 18.111, we have

$$z' = \frac{\partial z(x, b)}{\partial b}\delta b \tag{18.113}$$

which is called the variation of z with respect to b. Note that z' in Eq. 18.113 is also written as $z' = z'_{\delta b}$ to emphasize that the variation is taken specifically with respect to variable b.

Note that the coefficient of δb in Eq. 18.113, $\frac{\partial z}{\partial b}$, is the sensitivity of the state variable z—or more specifically, first-order sensitivity—which is equivalent to the derivative discussed in the discrete approach.

Because the direction of design change δb can be arbitrary (in this case, positive or negative), the variation of z stated in Eq. 18.113 must be linear with respect to δb. Again, for a simpler illustration, we assume a single design variable b. Mathematically, linearity of a function must have the following two important properties:

$$z'_{\delta b_1 + \delta b_2} = \frac{\partial z}{\partial b}(\delta b_1 + \delta b_2) = \frac{\partial z}{\partial b}\delta b_1 + \frac{\partial z}{\partial b}\delta b_2 = z'_{\delta b_1} + z'_{\delta b_2} \tag{18.114}$$

and

$$z'_{\alpha\delta b} = \frac{\partial z}{\partial b}(\alpha\delta b) = \alpha\frac{\partial z}{\partial b}\delta b = \alpha z'_{\delta b} \tag{18.115}$$

where α is a nonzero real number. Note that in Eq. 18.114, δb_1 and δb_2 represent two design perturbations of a single design variable b. For multiple design variables, δb_1 and δb_2 represent design perturbations of two different design variables or two different sets of design variables.

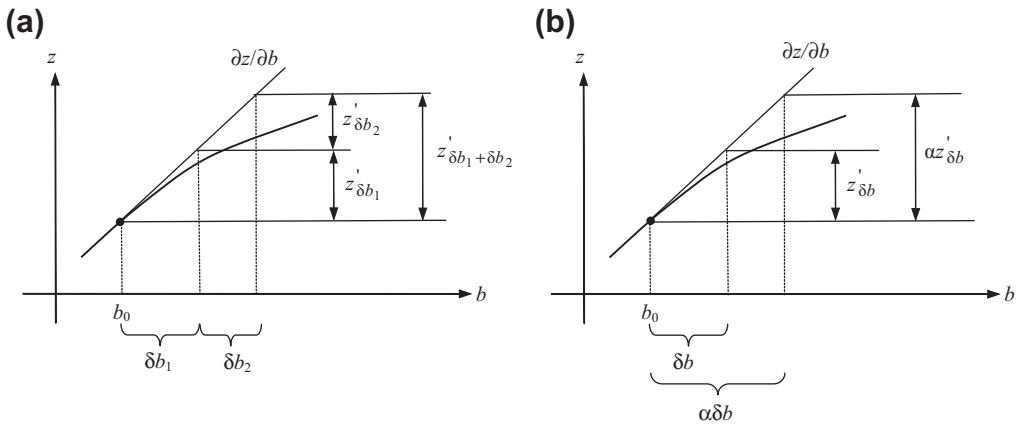


FIGURE 18.8

Illustration of the linearity properties of variations of the state variable z on design b: (a) $z'_{\delta b_1 + \delta b_2} = z'_{\delta b_1} + z'_{\delta b_2}$ and (b) $z'_{\alpha\delta b} = \alpha z'_{\delta b}$.

Geometrically, $\frac{\partial z}{\partial b}$ is the slope of the curve $z(b)$, as illustrated in Figure 18.8. As illustrated in Figure 18.8(a), $z'_{\delta b_1+\delta b_2}$ is the vertical distance of the slope $\partial z/\partial b$ times the design perturbation $\delta b_1 + \delta b_2$, which is equal to the sum of the slope multiplied by the individual design perturbations δb_1 and δb_2; that is, $z'_{\delta b_1} + z'_{\delta b_2}$. Similarly, the vertical distance of the slope $\partial z/\partial b$ times the design perturbation $\alpha\delta b$ is equal to the slope multiplied by the design perturbation δb times the constant α, as illustrated in Figure 18.8(b) Therefore, the variation of z stated in Eq. 18.113 is linear with respect to δb.

Note also that the variation and partial derivative are commutative, implying the order of the operations is interchangeable:

$$\left(z_{,i}\right)' = \left(z'\right)_{,i} \quad \text{or} \quad \left(\frac{\partial z}{\partial x_i}\right)' = \frac{\partial (z')}{\partial x_i} \tag{18.116}$$

and

$$\left(z_{,ij}\right)' = \left[\left(z_{,i}\right)'\right]_{,j} = \left(z'\right)_{,ij} \tag{18.117}$$

if the second derivative is involved.

EXAMPLE 18.9

A function f is defined as

$$f = f(b,z) = b + z^2 \tag{18.118}$$

We assume z is not a function of b. Calculate $\frac{\partial f}{\partial b}$, $\frac{\partial f}{\partial z}$, f', $f'_{\delta b}$, and $f'_{\delta z}$.

Solution

Taking derivatives of f with respect to b and z, respectively we have

$$\frac{\partial f}{\partial b} = 1, \quad \text{and} \quad \frac{\partial f}{\partial z} = 2z \tag{18.119}$$

Now we use definition of variation (Eq. 18.111) to calculate the variations f', $f'_{\delta b}$, and $f'_{\delta z}$.

$$f'_{\delta b}(b,z) \equiv \frac{\mathrm{d}}{\mathrm{d}\tau} f(b+\tau\delta b, z)\bigg|_{\tau=0} = \frac{\mathrm{d}}{\mathrm{d}\tau}\left(b+\tau\delta b + z^2\right)\bigg|_{\tau=0} = \delta b = f_{,b}\delta b \tag{18.120}$$

$$\begin{aligned} f'_{\delta z}(\mathrm{b},\mathrm{z}) &\equiv \frac{\mathrm{d}}{\mathrm{d}\tau} f(\mathrm{b},\mathrm{z}+\tau\delta\mathrm{z})\bigg|_{\tau=0} = \frac{\mathrm{d}}{\mathrm{d}\tau}\left(b+(z+\tau\delta\mathrm{z})^2\right)\bigg|_{\tau=0} \\ &= \frac{\mathrm{d}}{\mathrm{d}\tau}\left(b+z^2+2z\tau\delta\mathrm{z}+\tau^2\delta z^2\right)\bigg|_{\tau=0} = \left(2z\delta\mathrm{z}+2\tau\delta z^2\right)\bigg|_{\tau=0} = 2z\delta\mathrm{z} = f_{,z}\delta\mathrm{z} \end{aligned} \tag{18.121}$$

and

$$\begin{aligned} f' &\equiv \frac{\mathrm{d}}{\mathrm{d}\tau} f(b+\tau\delta b, z+\tau\delta z)\bigg|_{\tau=0} = \frac{\mathrm{d}}{\mathrm{d}\tau}\left(b+\tau\delta b+(z+\tau\delta\mathrm{z})^2\right)\bigg|_{\tau=0} \\ &= \frac{\mathrm{d}}{\mathrm{d}\tau}\left(b+\tau\delta b+z^2+2z\tau\delta z+\tau^2\delta z^2\right)\bigg|_{\tau=0} = \left(\delta b+2z\delta z+2\tau\delta z^2\right)\bigg|_{\tau=0} \\ &= \delta b + 2z\delta z = f_{,b}\delta b + f_{,z}\delta z = f'_{\delta b} + f'_{\delta z} \end{aligned} \tag{18.122}$$

EXAMPLE 18.10

A function f is defined as

$$f = f(b, z(b)) = b^2 + z(b)^2 \tag{18.123}$$

We assume z is a function of b. Calculate the variation of function f.

Solution

We use Eq. 18.111 to calculate the variation of f:

$$\begin{aligned} f'(b, z(b)) &\equiv \frac{\mathrm{d}}{\mathrm{d}\tau} f(b + \tau\delta b, z(b + \tau\delta b))\Big|_{\tau=0} = \frac{\mathrm{d}}{\mathrm{d}\tau}\left((b + \tau\delta b)^2 + z(b + \tau\delta b)^2\right)\Big|_{\tau=0} \\ &= \frac{\mathrm{d}}{\mathrm{d}\tau}\left((b + \tau\delta b)^2\right)\Big|_{\tau=0} + \frac{\mathrm{d}}{\mathrm{d}\tau}\left(z(b + \tau\delta b)^2\right)\Big|_{\tau=0} \\ &= 2b\delta b + 2z\frac{\mathrm{d}z(b + \tau\delta b)}{\mathrm{d}\tau}\Big|_{\tau=0} = f'_{\delta b} + 2z\frac{\partial z}{\partial b}\delta b \end{aligned} \tag{18.124}$$

where $f'_{\delta b} = 2b\delta b = f_{,b}\delta b = [f(b, \tilde{z})]'$, and $\frac{\mathrm{d}z(b+\tau\delta b)}{\mathrm{d}\tau}\Big|_{\tau=0} = \frac{\partial \tilde{z}}{\partial b}\delta b$. Note that $\tilde{z}$ implies that the dependence of z on b is suppressed in taking the variation. The first term of Eq. 18.124 $f'_{\delta b}$ represents the explicit dependence of f on variable b, and the second term represents the implicit dependence of f on variable b through $z(b)$.

EXAMPLE 18.11

A function f is defined as

$$f = f(b, z, z_{,b}) = b + z^2 + (z_{,b})^2 \tag{18.125}$$

We assume z is a function of b. Calculate the variation of function f.

Solution

We use Eq. 18.111 to calculate the variation f':

$$\begin{aligned} f'(b, z, z_{,b}) &\equiv \frac{\mathrm{d}}{\mathrm{d}\tau} f(b + \tau\delta b, z(b + \tau\delta b), z_{,b}(b + \tau\delta b))\Big|_{\tau=0} = \frac{\mathrm{d}}{\mathrm{d}\tau}\left(b + \tau\delta b + z(b + \tau\delta b)^2 + z_{,b}(b + \tau\delta b)^2\right)\Big|_{\tau=0} \\ &= \frac{\mathrm{d}}{\mathrm{d}\tau}(b + \tau\delta b)\Big|_{\tau=0} + \frac{\mathrm{d}}{\mathrm{d}\tau}\left(z(b + \tau\delta b)^2\right)\Big|_{\tau=0} + \frac{\mathrm{d}}{\mathrm{d}\tau}\left(z_{,b}(b + \tau\delta b)^2\right)\Big|_{\tau=0} \\ &= \delta b + 2z\frac{\mathrm{d}z(b + \tau\delta b)}{\mathrm{d}\tau}\Big|_{\tau=0} + 2z_{,b}\frac{\mathrm{d}z_{,b}(b + \tau\delta b)}{\mathrm{d}\tau}\Big|_{\tau=0} = f'_{\delta b} + 2z\frac{\partial z}{\partial b}\delta b + 2z_{,b}\frac{\partial z_{,b}}{\partial b}\delta b = f_{,b}\delta b + f_{,z}z' + f_{,z_{,b}}z'_{,b} \end{aligned} \tag{18.126}$$

18.4.2.2 Variations of the Energy Bilinear and Load Linear Forms

Now, we employ the definition of variation in Eq. 18.111 to the governing equation of general structure in variational form stated in Eq. 18.106:

$$a_b(\mathbf{z}, \overline{\mathbf{z}}) = \ell_b(\overline{\mathbf{z}}), \quad \forall\ \overline{\mathbf{z}} \in S \tag{18.127}$$

in which the subscript b is added to the energy bilinear and load linear forms to emphasize their dependence on design.

We assume a material or sizing design variable b. Employing the definition of variation in Eq. 18.111 to the energy bilinear form $a_b(\mathbf{z}, \overline{\mathbf{z}})$ on the left-hand side of Eq. 18.127, we have

$$[a_b(\mathbf{z}, \overline{\mathbf{z}})]' \equiv \frac{\mathrm{d}}{\mathrm{d}\tau}\left[a_b(\mathbf{z}(b + \delta b), \overline{\mathbf{z}})\right]\Big|_{\tau=0} = a'_{\delta b}(\mathbf{z}, \overline{\mathbf{z}}) + a_b(\mathbf{z}', \overline{\mathbf{z}}) \tag{18.128}$$

where

$$a'_{\delta b}(\mathbf{z}, \overline{\mathbf{z}}) = \frac{\mathrm{d}}{\mathrm{d}\tau}\left[a_b(\tilde{\mathbf{z}}, \overline{\mathbf{z}})\right]\Big|_{\tau=0} = [a_b(\tilde{\mathbf{z}}, \overline{\mathbf{z}})]' \tag{18.129}$$

which shows the explicit dependence of the energy bilinear form $a_b(\mathbf{z}, \overline{\mathbf{z}})$ on design variable b, which is similar to the first term on the right-hand side of Eq. 18.124 in Example 18.10—that is, $f'_{\delta b}$. The second term on the right-hand side, $a_b(\mathbf{z}', \overline{\mathbf{z}})$, represents the implicit dependence of $a_b(\mathbf{z}, \overline{\mathbf{z}})$ on design variables b through $\mathbf{z}'$, which is similar to the second term on the right-hand side of Eq. 18.124 in Example 18.10 that includes $\frac{\partial z}{\partial b}\delta b$.

By applying the definition of variation in Eq. 18.111 to the load linear form, on the right-hand side of Eq. 18.127, we have

$$[\ell_b(\overline{\mathbf{z}})]' \equiv \frac{\mathrm{d}}{\mathrm{d}\tau}\left[\ell_b(\overline{\mathbf{z}})\right]\Big|_{\tau=0} = \ell'_{\delta b}(\overline{\mathbf{z}}) \tag{18.130}$$

Hence, the sensitivity equation of the continuum approach of a static problem can be obtained as

$$a_b(\mathbf{z}', \overline{\mathbf{z}}) = -a'_{\delta b}(\mathbf{z}, \overline{\mathbf{z}}) + \ell'_{\delta b}(\overline{\mathbf{z}}), \quad \forall\ \overline{\mathbf{z}} \in S \tag{18.131}$$

The principle of virtual work states that the solution $\mathbf{z}'$ is sought in the function space S, such that Eq. 18.131 is satisfied for all $\overline{\mathbf{z}} \in S$. Solving $\mathbf{z}'$ using Eq. 18.131 is nothing but the direct differentiation method of the continuum approach. Similar to Eq. 18.106, the solutions $\mathbf{z}'$ of Eq. 18.131 exist and are unique.

Several important notes are stated below:

1. The terms on the left-hand sides of Eqs 18.131 and 18.127 are similar—that is, $a_b(\mathbf{z}', \overline{\mathbf{z}})$ and $a_b(\mathbf{z}, \overline{\mathbf{z}})$—except that displacement $\mathbf{z}$ in Eq. 18.127 is replaced by $\mathbf{z}'$ in Eq. 18.131.
2. $a_b(\mathbf{z}', \overline{\mathbf{z}})$ is bilinear because $a_b(\mathbf{z}, \overline{\mathbf{z}})$ is bilinear; that is, $a_b(\mathbf{z}', \overline{\mathbf{z}})$ is linear in $\mathbf{z}'$ and $\overline{\mathbf{z}}$, and $\ell'_{\delta b}(\overline{\mathbf{z}})$ is linear in $\overline{\mathbf{z}}$.
3. The right-hand side of Eq. 18.131 defines a pseudo-load (or fictitious load), which can be expressed explicitly in terms of the design variables. Thus, solving the sensitivity equation is the same as solving the original structural equation with a fictitious load per design variable.
4. $\mathbf{z}'$ is as smooth as $\mathbf{z}$; that is, both are in the Sobolev space $H^m(\Omega)$ and satisfy the essential boundary conditions of $\mathbf{z}' = 0$ on Γ_0.
5. For a static problem, $-a'_{\delta b}(\mathbf{z}, \overline{\mathbf{z}})$ is equivalent to $\frac{\partial \mathbf{K}}{\partial b}\mathbf{Z}$ in the discrete approach, and $\ell'_{\delta b}(\overline{\mathbf{z}})$ is equivalent to $\frac{\partial \mathbf{F}}{\partial b}$.

18.4.3 STATIC PROBLEMS

In this subsection, we discuss sensitivity analysis for static problems using the continuum approach. We use a beam example for illustration. We first derive the energy equation for the beam structure by specializing the general energy equations derived in Section 18.4.1 (instead of starting with a

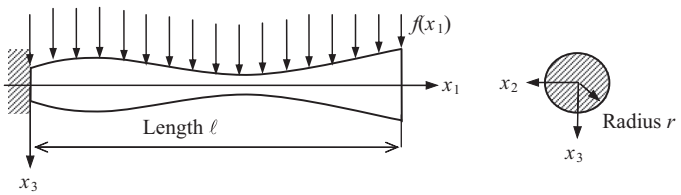


FIGURE 18.9

Cantilever beam example.

differential equation and carrying out integration by parts), and then introducing shape functions to formulate finite element matrix equations. We then discuss sensitivity analysis using both direct differentiation and adjoint variable methods.

Consider a cantilever beam with a circular cross-section under a distributed load $f(x_1)$, as shown in Figure 18.9. The differential governing equation of the beam can be written as

$$\left(EIz_{3,11}\right)_{,11} = f(x_1) \tag{18.132}$$

with essential boundary conditions

$$z_3(0) = z_{3,1}(0) = 0 \tag{18.133}$$

18.4.3.1 Energy Formulation

We derive the energy equation for the cantilever beam shown in Figure 18.9 by specializing the general energy equation of Eq. 18.105.

For a cantilever beam under bending, the displacement of a deformed beam shown in Figure 18.10(a) in the longitudinal (x_1-) direction is, as illustrated in Figure 18.10(b),

$$z_1 = -x_3 z_{3,1} \tag{18.134}$$

We assume a long beam (or so-called classical beam) where the transverse shear effect is neglected. As a result, the neutral axis is perpendicular to the section of the beam before and after deformation, as depicted in Figure 18.10(b). Note that the deformed configuration shown in Figure 18.10 shows $z_{3,1} > 0$ (following the sign convention we assume); therefore, a negative sign in front of z_1 is added in Eq. 18.134.

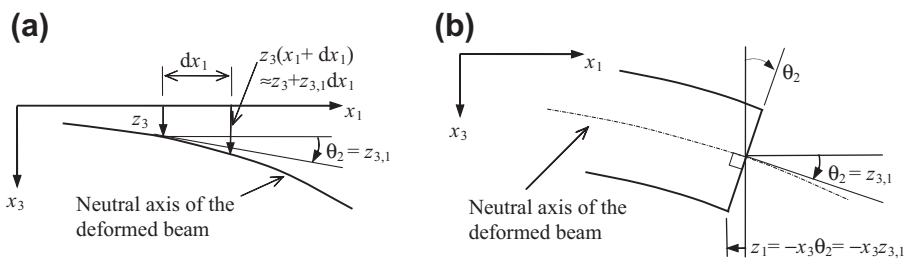


FIGURE 18.10

Deformed cantilever beam. (a) Neutral axis in a deformed configuration with major physical parameters. (b) Zoomed-in view at a given cross-section.

Therefore, the strain along the longitudinal direction, as illustrated in Figure 18.10(b), is

$$\varepsilon_{11} = z_{1,1} = \left(-x_3 z_{3,1}\right)_{,1} = -x_3 z_{3,11} \tag{18.135}$$

which is the only nonzero strain. Similarly, stress components consist of only bending stress σ_{11}, obtained as

$$\sigma_{11} = -E x_3 z_{3,11} \tag{18.136}$$

Using Eq. 18.105, we have

$$\begin{aligned}\int_0^\ell \int_A \sigma_{11}\bar{\varepsilon}_{11}\mathrm{d}A\mathrm{d}x &= \int_0^\ell \int_A \left(-E x_3 z_{3,11}\right)\left(-x_3 \bar{z}_{3,11}\right)\mathrm{d}A\mathrm{d}x_1 \\ &= \int_0^\ell \left(\int_A x_3^2 \mathrm{d}A\right) E z_{3,11}\bar{z}_{3,11}\mathrm{d}x_1 = \int_0^\ell EI z_{3,11}\bar{z}_{3,11}\mathrm{d}x_1 = \int_0^\ell f(x_1)\bar{z}_3\mathrm{d}x_1\end{aligned} \tag{18.137}$$

Note that the traction force T is zero for the beam example shown in Figure 18.9. The principle of virtual work states that the solution z_3 is sought in the space of kinematically admissible virtual displacement S such that Eq. 18.137 is satisfied for all $\bar{z}_3 \in S$. The kinematically admissible virtual displacement S for the cantilever beam is defined as

$$S = \left\{ z_3 \in H^2[0,\ell] \,\middle|\, z_3(0) = z_{3,1}(0) = 0 \right\} \tag{18.138}$$

where H^2 is the Sobolev space of order 2, implying the second-order derivative of the solution z_3 must be integrable.

EXAMPLE 18.12

Derive the energy equation for the cantilever beam shown in Figure 18.9 using integration by parts from the differential equations of Eqs 18.132 and 18.133.

Solution

We introduce the virtual displacement $\bar{z}_3$ that satisfies the essential boundary conditions of Eq. 18.133, and multiply it to both sizes of Eq. 18.132:

$$\int_0^\ell \left(EIz_{3,11}\right)_{,11}\bar{z}_3\mathrm{d}x_1 = \int_0^\ell f(x_1)\bar{z}_3\mathrm{d}x_1 \tag{18.139}$$

We carry out integration by parts twice for Eq. 18.139 to obtain

$$\left(EIz_{3,11}\right)_{,1}\bar{z}_3\Big|_0^\ell - EIz_{3,11}\bar{z}_{3,1}\Big|_0^\ell + \int_0^\ell EIz_{3,11}\bar{z}_{3,11}\mathrm{d}x_1 = \int_0^\ell f(x_1)\bar{z}_3\mathrm{d}x_1 \tag{18.140}$$

Equation (18.140) is called the variational identity and is true for a beam with any boundary condition.

Because $\bar{z}_3(0) = \bar{z}_{3,1}(0) = 0$ and the bending moment and transverse shear force at the tip are zero—that is, $EIz_{3,11}(\ell) = 0$, $\left(EIz_{3,11}\right)_{,1}(\ell) = 0$—the first two terms on the left-hand side of Eq. 18.140 vanish. Then, Eq. 18.140 becomes

$$\int_0^\ell EIz_{3,11}\bar{z}_{3,11}\mathrm{d}x_1 = \int_0^\ell f(x_1)\bar{z}_3\mathrm{d}x_1, \quad \forall \bar{z}_3 \in S \tag{18.141}$$

where S is the space of kinematically admissible virtual displacement defined in Eq. 18.138.

18.4.3.2 Finite Element Discretization

For a beam element under bending, as shown in Figure 18.11(b), there are two degrees of freedom at each node: the displacement z_3 and rotation angle θ. For simplicity, we assume constant area A and constant modulus E for the cantilever beam (see Figure 18.11(a)). Also, the distributed load is assumed to be constant—that is, $f(x_1) = q$. In FEA, a cubic shape function is employed for a (classical) beam element. For a cantilever beam with a point load at the tip, one beam element is sufficient to provide the exact solution because the exact solution is a cubic function of location x_1. However, for a beam with a uniformly distributed load q, the exact displacement solution is a fourth-order function of x_1. A finite element with cubic shape functions will not give an exact solution, except at nodes.

Using cubic shape functions, the displacement function $z_3(x)$ in the beam element can be interpolated as

$$z_3(x) = \mathbf{N}^{\mathrm{T}}\mathbf{Z} = [N_1\ N_2\ N_3\ N_4]\begin{bmatrix} z_{3_i} \\ \theta_i \\ z_{3_j} \\ \theta_j \end{bmatrix} \tag{18.142}$$

where

$$\begin{aligned} N_1 &= 1 - \frac{3x^2}{\ell^2} + \frac{2x^3}{\ell^3}, \quad N_2 = x - \frac{2x^2}{\ell} + \frac{x^3}{\ell^2}, \\ N_3 &= \frac{3x^2}{\ell^2} - \frac{2x^3}{\ell^3}, \quad N_4 = -\frac{x^2}{\ell} + \frac{x^3}{\ell^2} \end{aligned} \tag{18.143}$$

Hence, the second derivative of the displacement can be interpolated as

$$z_{3,11} = \left(\mathbf{N}^{\mathrm{T}}\mathbf{Z}\right)_{,11} = \mathbf{N}^{\mathrm{T}}_{,11}\mathbf{Z} = \left[-\frac{6}{\ell^2} + \frac{12x}{\ell^3} - \frac{4}{\ell} + \frac{6x}{\ell^2}\ \ \frac{6}{\ell^2} - \frac{12x}{\ell^3} - \frac{2}{\ell} + \frac{6x}{\ell^2}\right]\begin{bmatrix} z_{3_i} \\ \theta_i \\ z_{3_j} \\ \theta_j \end{bmatrix}$$

Similarly, $\overline{z}_{3,11} = (\mathbf{N}^{\mathrm{T}}\overline{\mathbf{Z}})_{,11} = \mathbf{N}^{\mathrm{T}}_{,11}\overline{\mathbf{Z}}$.

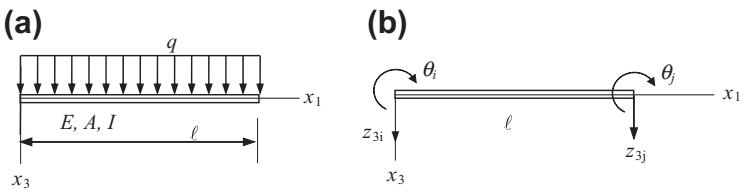


FIGURE 18.11

A simple cantilever beam example. (a) A beam with a uniformly distributed load q. (b) A beam element with displacement and rotation degrees of freedom.

Note that the shape functions of Eq. 18.143 are derived in accordance with the sign conventions shown in Figure 18.11(b). Also, to simplify the notation, we omitted the subscript 1 of x_1 in the above equations and those that follow.

Plug $z_{3,11}$ and $\overline{z}_{3,11}$ into the energy bilinear form of Eq. 18.137 and carry out the integrations, which yields

$$\int_0^\ell EIz_{3,11}\overline{z}_{3,11}\mathrm{d}x = \overline{\mathbf{Z}}^{\mathrm{T}}\mathbf{K}_{\mathrm{g}}\mathbf{Z} \tag{18.144}$$

Here, $\mathbf{K}_{\mathrm{g}}$ is the global stiffness matrix:

$$\mathbf{K}_{\mathrm{g}} = \frac{EI}{\ell^3}\begin{bmatrix} 12 & 6\ell & -12 & 6\ell \\ 6\ell & 4\ell^2 & -6\ell & 2\ell^2 \\ -12 & -6\ell & 12 & -6\ell \\ 6\ell & 2\ell^2 & -6\ell & 4\ell^2 \end{bmatrix}$$

and

$$\int_0^\ell f(x)\overline{\mathbf{z}}\mathrm{d}x = \overline{\mathbf{Z}}^{\mathrm{T}}\int_0^\ell q\mathbf{N}\mathrm{d}x = \overline{\mathbf{Z}}^{\mathrm{T}}\begin{bmatrix} \frac{1}{2}q\ell \\ \frac{1}{12}q\ell^2 \\ \frac{1}{2}q\ell \\ -\frac{1}{12}q\ell^2 \end{bmatrix} = \overline{\mathbf{Z}}^{\mathrm{T}}\mathbf{F}_{\mathrm{g}} \tag{18.145}$$

Here, $\mathbf{F}_{\mathrm{g}}$ is the global force vector. Equation (18.137) becomes $\overline{\mathbf{Z}}^{\mathrm{T}}\mathbf{K}_{\mathrm{g}}\mathbf{Z} = \overline{\mathbf{Z}}^{\mathrm{T}}\mathbf{F}_{\mathrm{g}}$. Because $\overline{\mathbf{Z}}$ is a virtual displacement that is arbitrary in space S, it can be removed. Hence, we have

$$\mathbf{K}_{\mathrm{g}}\mathbf{Z} = \mathbf{F}_{\mathrm{g}}$$

or

$$\frac{EI}{\ell^3}\begin{bmatrix} 12 & 6\ell & -12 & 6\ell \\ 6\ell & 4\ell^2 & -6\ell & 2\ell^2 \\ -12 & -6\ell & 12 & -6\ell \\ 6\ell & 2\ell^2 & -6\ell & 4\ell^2 \end{bmatrix}\begin{bmatrix} z_{3_i} \\ \theta_i \\ z_{3_j} \\ \theta_j \end{bmatrix} = \begin{bmatrix} \frac{1}{2}q\ell \\ \frac{1}{12}q\ell^2 \\ \frac{1}{2}q\ell \\ -\frac{1}{12}q\ell^2 \end{bmatrix} \tag{18.146}$$

Impose the boundary conditions. In this case, $z_{3i} = \theta_i = 0$. The reduced matrix equations become $\mathbf{KZ} = \mathbf{F}$, or

$$\frac{EI}{\ell^3}\begin{bmatrix} 12 & -6\ell \\ -6\ell & 4\ell^2 \end{bmatrix}\begin{bmatrix} z_{3_j} \\ \theta_j \end{bmatrix} = \begin{bmatrix} \frac{1}{2}q\ell \\ -\frac{1}{12}q\ell^2 \end{bmatrix} \tag{18.147}$$

Inverse the matrix and solve for the nodal point displacements, yielding

$$\begin{bmatrix} z_{3_j} \\ \theta_j \end{bmatrix} = \frac{\ell^3}{EI}\begin{bmatrix} 12 & -6\ell \\ -6\ell & 4\ell^2 \end{bmatrix}^{-1}\begin{bmatrix} \frac{1}{2}q\ell \\ -\frac{1}{12}q\ell^2 \end{bmatrix} = \frac{\ell}{12EI}\begin{bmatrix} 4\ell^2 & 6\ell \\ 6\ell & 12 \end{bmatrix}\begin{bmatrix} \frac{1}{2}q\ell \\ -\frac{1}{12}q\ell^2 \end{bmatrix} = \begin{bmatrix} \frac{q\ell^4}{8EI} \\ \frac{q\ell^3}{6EI} \end{bmatrix} \tag{18.148}$$

Note that from the strength of materials, the exact solution for the cantilever beam with a uniformly distributed load q is

$$z_3^{\text{Exact}}(x) = \frac{q}{24EI}\left(6x^2\ell^2 - 4x^3\ell + x^4\right) \tag{18.149}$$

The finite element solutions in Eq. 18.148 match those of the exact solutions at the ends (in this case, finite element nodes), both the displacement and rotation angles.

The finite element solution for displacement $z_3(x)$ for the entire element can then be obtained from Eq. 18.142 as

$$z_3(x) = \mathbf{N}^{\mathrm{T}}\mathbf{Z} = [N_3 N_4]\begin{bmatrix} z_{3_j} \\ \theta_j \end{bmatrix} = \begin{bmatrix} \frac{3x^2}{\ell^2} - \frac{2x^3}{\ell^3}, & -\frac{x^2}{\ell} + \frac{x^3}{\ell^2} \end{bmatrix}\begin{bmatrix} \frac{q\ell^4}{8EI} \\ \frac{q\ell^3}{6EI} \end{bmatrix} = \frac{q\ell}{24EI}\left(5x^2\ell - 2x^3\right) \tag{18.150}$$

where $z_3(x)$ represents the beam displacements due to the distributed load q, which is a cubic function because cubic shape functions were employed for the finite element formulation.

Apparently, the FEM does not give the exact solution because the element shape function is a cubic function, and the exact solution is a fourth-order polynomial function. When taking the second derivatives of $z_3(x)$ and $z_3^{\text{Exact}}(x)$ from Eqs 18.150 and 18.149, respectively, we have $z_{3,11}(x) = \frac{q\ell}{12EI}(5\ell - 6x)$ and $z_{3,11}^{\text{Exact}}(x) = \frac{q}{2EI}(\ell^2 - 2x\ell + x^2)$. At $x = \ell$, we have $z_{3,11}(\ell) = -\frac{q\ell^2}{12EI} \neq 0$ and $z_{3,11}^{\text{Exact}}(\ell) = 0$. Note that $EIz_{3,11}(x)$ is the bending moment in the beam. At the tip ($x = \ell$), the bending moment of the cantilever beam should be zero. It is apparent that the exact solution $z_3^{\text{Exact}}(x)$ captures this important behavior; however, the approximate solution obtained from FEA does not. In general, when more elements are employed for the cantilever beam (i.e., by dividing the beam into more elements of smaller size), the finite element solution eventually approaches the exact solution.

18.4.3.3 Direct Differentiation Method of the Continuum-Discrete Approach

For linear static problems, we focus on displacement, stress, and compliance performance measures. We have discussed bars in Section 18.3. In this section, we use beams for illustration.

We assume a rectangular cross-section for the cantilever beam shown in Figure 18.11, with width w and height h. We use the direct differentiation method of the continuum-discrete approach to calculate the sensitivity of the bending stress σ with the height h as the design variable.

Taking the variation of the energy equation (Eq. 18.137) with respect to the height h, we have

$$\int_0^\ell EIz'_{3,11}\bar{z}_{3,11}\mathrm{d}x = -\int_0^\ell EI' z_{3,11}\bar{z}_{3,11}\mathrm{d}x, \quad \forall \bar{\mathbf{z}} \in S \tag{18.151}$$

in which $I = \frac{wh^3}{12}$, $I' = \delta I = \frac{\partial I}{\partial h}\delta h = \frac{3I}{h}\delta h$, and $z'_3 = \frac{\partial z_3}{\partial h}\delta h$. From the principle of virtual work, the solution of Eq. 18.151—that is, z'_3—belongs to the space S.

Using the same shape functions as in Eq. 18.143 to discretize the sensitivity equation (Eq. 18.151), we have on the left-hand side

$$\int_0^\ell EIz'_{3,11}\bar{z}_{3,11}\mathrm{d}x = \overline{\mathbf{Z}}^{\mathrm{T}}\mathbf{K}_{\mathrm{g}}\mathbf{Z}' \tag{18.152}$$

where

$$\mathbf{Z}' = \frac{\partial \mathbf{Z}}{\partial h}\delta h = \left[\frac{\partial z_{3_i}}{\partial h}\delta h \quad \frac{\partial \theta_i}{\partial h}\delta h \quad \frac{\partial z_{3_j}}{\partial h}\delta h \quad \frac{\partial \theta_j}{\partial h}\delta h\right]^{\mathrm{T}} \tag{18.153}$$

On the right-hand side of Eq. 18.151, we have

$$-\int_0^\ell EI' z_{3,11}\bar{z}_{3,11}\mathrm{d}x = -\frac{3EI}{h}\overline{Z}^T \int_0^\ell \begin{bmatrix} \frac{-6}{\ell^2} + 12\frac{x}{\ell^3} \\ \frac{-4}{\ell} + 6\frac{x}{\ell^2} \\ \frac{6}{\ell^2} - 12\frac{x}{\ell^3} \\ \frac{-2}{\ell} + 6\frac{x}{\ell^2} \end{bmatrix} \left[\frac{q\ell}{12EI}(5\ell - 6x)\right]\mathrm{d}x\delta h = \frac{q\ell}{4h}\begin{bmatrix} -18 \\ 5\ell \\ -6 \\ \ell \end{bmatrix}\delta h \tag{18.154}$$

By imposing the boundary conditions and removing the virtual displacement $\overline{\mathbf{Z}}$ and δh from both sides, we have $\mathbf{K}\mathbf{Z}' = \mathbf{F}_{\mathrm{fic}}$:

$$\frac{EI}{\ell^3}\begin{bmatrix} 12 & -6\ell \\ -6\ell & 4\ell^2 \end{bmatrix}\begin{bmatrix} \frac{\partial z_{3_j}}{\partial h} \\ \frac{\partial \theta_j}{\partial h} \end{bmatrix} = \frac{q\ell}{4h}\begin{bmatrix} -6 \\ \ell \end{bmatrix} \tag{18.155}$$

The right-hand side becomes a vector of a fictitious load:

$$\mathbf{F}_{\text{fic}} = \frac{q\ell}{4h}\begin{bmatrix} -6 \\ \ell \end{bmatrix} \tag{18.156}$$

By solving the sensitivity vector in the same way as solving for $\mathbf{Z}$ of the structure (see Eq. 18.148), we have

$$\begin{bmatrix} \dfrac{\partial z_{3_j}}{\partial h} \\ \dfrac{\partial \theta_j}{\partial h} \end{bmatrix} = \mathbf{K}^{-1}\mathbf{F}_{\text{fic}} = \frac{\ell}{12EI}\begin{bmatrix} 4\ell^2 & 6\ell \\ 6\ell & 12 \end{bmatrix}\frac{q\ell}{4h}\begin{bmatrix} -6 \\ \ell \end{bmatrix} = \begin{bmatrix} -\dfrac{3q\ell^4}{8EIh} \\ -\dfrac{q\ell^3}{2EIh} \end{bmatrix} \tag{18.157}$$

18.4.3.4 Sensitivity of the Bending Stress

Now, we calculate the sensitivity of the bending stress σ_{11} with respect to the height h of the beam cross-section. The bending stress σ_{11} can be calculated using the displacement results obtained from FEA as $\sigma_{11}=-Ex_3z_{3,11}$, and the maximum tensile stress is at the top fiber ($x_3 = -h/2$) of the beam:

$$\sigma_{11} = E\frac{h}{2}z_{3,11} \tag{18.158}$$

Note that we follow the sign conventions shown in Figure 18.12 for the bending moment and stress.

The sensitivity of the maximum bending stress in tensile σ_{11} with respect to height h can be computed by taking the derivative of Eq. 18.158 as

$$\frac{\partial \sigma_{11}}{\partial h} = \frac{E}{2}z_{3,11} + \frac{Eh}{2}\frac{\partial z_{3,11}}{\partial h} \tag{18.159}$$

Here, using finite element solution $z_{3,11}$ can be obtained by taking the second derivative of z_3 obtained in Eq. 18.150 with respect to x, as

$$z_{3,11}(x) = \frac{q\ell}{12EI}(5\ell - 6x)$$

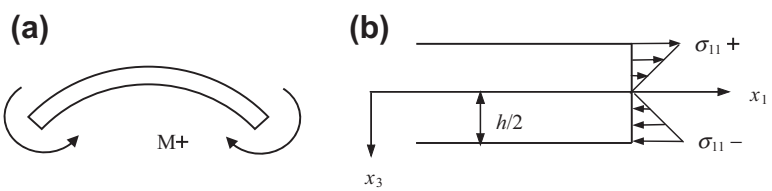


FIGURE 18.12

Sign conventions of beam stress. (a) Positive bending moment. (b) Positive stress in tensile and negative stress in compression.

The sensitivity of $z_{3,11}$ can be interpolated using element shape functions, similar to that of displacement z_3 in Eq. 18.150, as

$$\frac{\partial z_{3,11}(x)}{\partial h} = \mathbf{N}_{,11}^{\mathrm{T}} \frac{\partial \mathbf{Z}}{\partial h} = \begin{bmatrix} N_{3,11} & N_{4,11} \end{bmatrix} \begin{bmatrix} \dfrac{\partial z_{3_j}}{\partial h} \\ \dfrac{\partial \theta_j}{\partial h} \end{bmatrix} = \left[\frac{6}{\ell^2} - \frac{12x}{\ell^3}, -\frac{2}{\ell} + \frac{6x}{\ell^2} \right] \begin{bmatrix} -\dfrac{3q\ell^4}{8EIh} \\ -\dfrac{q\ell^3}{2EIh} \end{bmatrix} = \frac{q\ell}{4EIh}(6x - 5\ell)$$

Hence, Eq. 18.159 becomes

$$\frac{\partial \sigma_{11}}{\partial h} = \frac{q\ell}{24I}(5\ell - 6x) + \frac{q\ell}{8I}(6x - 5\ell) = -\frac{q\ell}{12I}(5\ell - 6x) \tag{18.160}$$

At the root when $x = 0$, we have

$$\frac{\partial \sigma_{11}}{\partial h} = -\frac{5q\ell^2}{12I} < 0 \tag{18.161}$$

This implies that increasing the height h will decrease the maximum bending stress in tensile at the top fiber of the beam.

Note that in theory, as mentioned earlier, the stress sensitivity at the tip $x = \ell$ is zero because the bending moment—and hence, the bending stress—is zero at the tip. However, as indicated in Eq. 18.160, this is not the case because the solution we used for evaluating the sensitivity information is obtained from FEA. As we are aware, the finite element solution is not exact.

If we use the exact solution to calculate the sensitivity using the continuum-analytical method, the stress sensitivity is exact. We illustrate this point in the next example.

EXAMPLE 18.13

The exact solution of the cantilever beam shown in Figure 18.11(a) was given in Eq. 18.149 and is restated below:

$$z_3^{\mathrm{Exact}}(x) = \frac{q}{24EI}\left(6x^2\ell^2 - 4x^3\ell + x^4\right) \tag{18.162}$$

We use the exact solution to calculate the sensitivity of the maximum bending stress of the beam defined in Eq. 18.158 using the continuum formulation. Because no finite element discretization is involved, we are exercising the continuum-analytical approach in this example.

Solution

We first carry out two integrations by parts on Eq. 18.151 to factor out the virtual displacement $\bar{z}$. We are getting back to the differential equation so that we can solve for z_3' when a fictitious load is calculated. The left-hand side of Eq. 18.151 becomes

$$\int_0^\ell EIz'_{3,11}\bar{z}_{3,11}\mathrm{d}x = EIz'_{3,11}\bar{z}_{3,1}\Big|_0^\ell - EIz'_{3,111}\bar{z}_3\Big|_0^\ell + \int_0^\ell EIz'_{3,1111}\bar{z}_3\mathrm{d}x \tag{18.163}$$

The first two terms vanish because $\bar{z}_3(0) = \bar{z}_{3,1}(0) = 0$ and z_3' belongs to the same space as z_3, in which $EIz_{3,11}(\ell) = 0$, $(EIz_{3,11})_{,1}(\ell) = 0$, as discussed earlier in this subsection. You may come back to check if $EIz'_{3,11}(\ell) = 0$ and $(EIz'_{3,11})_{,1}(\ell) = 0$ once we obtain z_3'.

Continued

EXAMPLE 18.13—cont'd

The right-hand side of Eq. 18.151 becomes

$$-\int_0^\ell EI' z_{3,11}\bar{z}_{3,11}\mathrm{d}x = -EI' z_{3,11}\bar{z}_{3,1}\Big|_0^\ell + EI' z_{3,111}\bar{z}_3\Big|_0^\ell - \int_0^\ell EI' z_{3,1111}\bar{z}_3\mathrm{d}x \tag{18.164}$$

Again, the first two terms of Eq. 18.164 vanish. We remove the integral and delete the virtual displacement $\bar{z}_3$ on both sides of the equation to obtain the sensitivity equation in differential form as

$$EIz'_{3,1111} = -EI' z_{3,1111} = -\frac{3EI}{h} z_{3,1111}\delta h = -\frac{3EI}{h}\frac{q}{EI}\delta h = -\frac{3q}{h}\delta h \tag{18.165}$$

Hence, the solution of Eq. 18.165 can be obtained as

$$z'_3(x) = -\frac{3q}{24EIh}\left(6x^2\ell^2 - 4x^3\ell + x^4\right)\delta h \tag{18.166}$$

Its second derivative with respect to x is

$$z'_{3,11}(x) = -\frac{3q}{2EIh}\left(\ell^2 - 2\ell x + x^2\right)\delta h \tag{18.167}$$

At the tip, $z'_{3,11}(\ell) = 0$ and $z'_{3,111}(\ell) = 0$ as expected. From Eq. 18.159, the sensitivity of bending stress σ_{11} is

$$\frac{\partial\sigma_{11}}{\partial h} = \frac{E}{2}z_{3,11} + \frac{Eh}{2}\frac{\partial z_{3,11}}{\partial h} = \frac{q}{4I}\left(\ell^2 - 2x\ell + x^2\right) - \frac{3q}{4I}\left(\ell^2 - 2\ell x + x^2\right) = -\frac{q}{2I}(\ell - x)^2 \tag{18.168}$$

Note that $\frac{\partial\sigma_{11}}{\partial h} < 0$ except at the tip, implying that increasing the height h reduces the maximum bending stress. At the tip, $\frac{\partial\sigma_{11}}{\partial h} = 0$ as expected. The first term of Eq. 18.168 represents the explicit dependence of the stress on the height variable, which is always positive (except at the tip) because increasing h—that is, raising the stress measurement along the height of the beam cross-section—increases bending stress.

Another point worth mentioning is that the stress sensitivity shown in Eq. 18.159 assumes that the stress measurement point is changing as the height h varies. If we assume the stress measurement point stays, the first term of Eq. 18.159 (and Eq. 18.168) vanishes. Therefore, the stress sensitivity becomes

$$\frac{\partial\sigma_{11}}{\partial h} = \frac{Eh}{2}\frac{\partial z_{3,11}}{\partial h} = -\frac{3q}{4I}\left(\ell^2 - 2\ell x + x^2\right) = -\frac{3q}{4I}(\ell - x)^2 \tag{18.169}$$

This implies that the stress decreases more as the height h increases if the measurement point does not move together with the height h.

18.4.3.5 Adjoint Variable Method of the Continuum-Discrete Approach

We discuss the adjoint variable method of the continuum-discrete approach for a general performance measure, and then use the stress performance measure of the cantilever beam example for illustration.

In the continuum approach, the performance measure is expressed in an integral form as

$$\psi = \int_\Omega g(\mathbf{z}(\mathbf{b}), \nabla\mathbf{z}(\mathbf{b}); \mathbf{b})\mathrm{d}\Omega \tag{18.170}$$

where Ω is the structural domain. For a one-dimensional problem such as the cantilever beam, $\Omega = [0,\ell]$. The displacement $\mathbf{z}$ is a function of design variables $\mathbf{b}$, and $\nabla\mathbf{z}$ is the gradient of $\mathbf{z}$ with respect to spatial variables $\mathbf{x}$. For simplicity, we assume a single design variable b. Note that the

displacement $\mathbf{z}$ depends on both the spatial variable $\mathbf{x}$ and design variable b. For a one-dimensional problem, such as the cantilever beam in bending, vector $\mathbf{z}$ becomes a scalar quantity z_3, and $\mathbf{x}$ becomes x_1.

Taking the variation of the performance measure ψ with respect to design variable b, we have

$$\begin{aligned}\psi' &\equiv \frac{\mathrm{d}}{\mathrm{d}\tau}\left[\int_\Omega g(\mathbf{z}(\mathbf{x}, b+\tau\delta b), \nabla\mathbf{z}(\mathbf{x}, b+\tau\delta b), b+\tau\delta b)\mathrm{d}\Omega\right]\Bigg|_{\tau=0} \\ &= \int_\Omega \left[g_{,\mathbf{z}}\mathbf{z}' + g_{,\nabla\mathbf{z}}\nabla\mathbf{z}' + g_{,b}\delta b\right]\mathrm{d}\Omega\end{aligned} \tag{18.171}$$

Here, ∇ is the gradient operator, and is defined as

$$\nabla \equiv \begin{bmatrix} \dfrac{\partial}{\partial x_1} \\ \dfrac{\partial}{\partial x_2} \\ \dfrac{\partial}{\partial x_3} \end{bmatrix} \tag{18.172}$$

for three-dimensional structures. If we simplify the formulation by considering only one-dimensional problems, we have $\nabla = \frac{\partial}{\partial x_1}$ and Eq. 18.171 becomes

$$\psi' = \int_0^\ell \left[g_{,\mathbf{z}}\mathbf{z}' + g_{,\mathbf{z}_{,1}}\mathbf{z}'_{,1} + g_{,b}\delta b\right]\mathrm{d}x \tag{18.173}$$

In Eq. 18.171 (and Eq. 18.173), the third term in the integrand $g_{,b}\delta b$ represents the explicit dependence of g in design b, and the first two terms are implicit terms that need to be substituted by introducing the adjoint equation.

Note that some terms in the integrand of Eq. 18.173 are zero depending on the type of performance measures. For example, for a displacement measure, $g_{,b}\delta b$ is zero because no explicit dependence exists, and $g_{,\mathbf{z}_{,1}}$ is zero because no derivative is involved in a displacement measure. For a stress performance measure, as seen in previous examples (e.g., Example 18.13), $g_{,b}\delta b$ is nonzero, as well as $g_{,\mathbf{z}_{,1}}$. For beams in bending, we have $g_{,z_{3,11}}$ instead of $g_{,\mathbf{z}_{,1}}$.

The adjoint equation is written as follows, replacing the terms involved in variations in Eq. 18.173:

$$a_b(\boldsymbol{\lambda}, \overline{\boldsymbol{\lambda}}) = \int_0^\ell \left[g_{,\mathbf{z}}\overline{\boldsymbol{\lambda}} + g_{,\mathbf{z}_{,1}}\overline{\boldsymbol{\lambda}}_{,1}\right]\mathrm{d}x, \quad \forall\, \overline{\boldsymbol{\lambda}} \in S \tag{18.174}$$

Note that the left-hand side of Eq. 18.174 is the energy bilinear form in terms of adjoint response $\boldsymbol{\lambda}$ and virtual adjoint response $\overline{\boldsymbol{\lambda}}$. The adjoint response can be obtained in the same way as the structural response $\mathbf{z}$ (e.g., using finite element equations), as long as the load vector, called adjoint load, on the right-hand side can be calculated.

We first evaluate Eq. 18.174 at $\overline{\boldsymbol{\lambda}} = \mathbf{z}'$:

$$a_b(\boldsymbol{\lambda}, \mathbf{z}') = \int_0^{\ell} \left[\mathrm{g}_{,\mathbf{z}} \mathbf{z}' + \mathrm{g}_{,\mathbf{z}_{,1}} \mathbf{z}'_{,1} \right] \mathrm{d}x \ , \ \ \forall \mathbf{z}' \in \mathrm{S} \tag{18.175}$$

Recall the sensitivity equation of the direct differentiation method in Eq. 18.131. We evaluate Eq. 18.131 at $\overline{\mathbf{z}} = \boldsymbol{\lambda}$:

$$a_b(\mathbf{z}', \boldsymbol{\lambda}) = -a'_{\delta b}(\mathbf{z}, \boldsymbol{\lambda}) + \ell'_{\delta b}(\boldsymbol{\lambda}) \tag{18.176}$$

Because the energy bilinear form is symmetric—that is, $a_b(\boldsymbol{\lambda}, \mathbf{z}') = a_b(\mathbf{z}', \boldsymbol{\lambda})$, Eq. 18.175 equals Eq. 18.176. Therefore,

$$\int_0^{\ell} \left[g_{,\mathbf{z}} \mathbf{z}' + g_{,\mathbf{z}_{,1}} \mathbf{z}'_{,1} \right] \mathrm{d}x = -a'_{\delta b}(\mathbf{z}, \boldsymbol{\lambda}) + \ell'_{\delta b}(\boldsymbol{\lambda}) \tag{18.177}$$

Bringing Eq. 18.177 back to Eq. 18.173, we have

$$\psi' = \int_0^{\ell} g_{,b} \delta b \mathrm{d}x - a'_{\delta b}(\mathbf{z}, \boldsymbol{\lambda}) + \ell'_{\delta b}(\boldsymbol{\lambda}) \tag{18.178}$$

This can be calculated once the original structural response $\mathbf{z}$ and adjoint response $\boldsymbol{\lambda}$ are available. Note that for beams, we have

$$a'_{\delta b}(\mathbf{z}, \boldsymbol{\lambda}) = \int_0^{\ell} EIz_{3,11} \lambda_{3,11} \mathrm{d}x \tag{18.179}$$

and

$$\ell'_{\delta b}(\boldsymbol{\lambda}) = \int_0^{\ell} f \lambda_3 \mathrm{d}x \tag{18.180}$$

EXAMPLE 18.14

The stress measure at the top fiber of the cantilever beam shown in Figure 18.11(a) at the root ($x = 0$) is written in an integral form as

$$\psi = \sigma_{11}(0) = E \frac{h}{2} z_{3,11}(0) = \int_0^{\ell} \left[E \frac{h}{2} z_{3,11}(x) \widehat{\delta}(x - 0) \right] \mathrm{d}x \tag{18.181}$$

in which $\widehat{\delta}(x-0)$ is a direct delta function defined as

$$\widehat{\delta}(x - 0) = \begin{cases} \infty, \ x = 0 \\ 0, \ x \neq 0 \end{cases} \tag{18.182}$$

EXAMPLE 18.14—cont'd

Calculate the sensitivity of the stress measure with respect to the height design variable h using the adjoint variable method of the continuum-discrete approach.

Solution

Taking the variation of the performance measure ψ in Eq. 18.181, we have

$$\begin{aligned}\psi' &= \int_0^\ell \left[E\frac{\delta h}{2} z_{3,11}(x)\widehat{\delta}(x-0) + E\frac{h}{2} z'_{3,11}(x)\widehat{\delta}(x-0)\right]\mathrm{d}x \\ &= E\frac{\delta h}{2} z_{3,11}(0) + \int_0^\ell \left[E\frac{h}{2} z'_{3,11}(x)\widehat{\delta}(x-0)\right]\mathrm{d}x\end{aligned} \tag{18.183}$$

in which the first term on the right-hand side shows the explicit dependence of the performance measure on design—that is, $\int_0^\ell g_{,b}\delta b\mathrm{d}x$ in Eq. 18.173—which is straightforward to solve. The second term involves z'_3, which must be solved by introducing the adjoint structure, defined as

$$a_b(\lambda, \overline{\lambda}) = \int_0^\ell \left[E\frac{h}{2}\overline{\lambda}_{3,11}(x)\widehat{\delta}(x-0)\right]\mathrm{d}x \tag{18.184}$$

Discretizing Eq. 18.184 in the same way as the original structure using the shape functions in Eq. 18.143, the left-hand side becomes

$$a_b(\lambda, \overline{\lambda}) = \int_0^\ell EI\lambda_{3,11}\overline{\lambda}_{3,11}\mathrm{d}x = \overline{\boldsymbol{\lambda}}_{\mathrm{g}}^{\mathrm{T}}\mathbf{K}_{\mathrm{g}}\boldsymbol{\lambda}_{\mathrm{g}} \tag{18.185}$$

and the right-hand side is

$$\begin{aligned}&\overline{\boldsymbol{\lambda}}_{\mathrm{g}}^{\mathrm{T}}\int_0^\ell \left[E\frac{h}{2}\mathbf{N}_{,11}(x)\widehat{\delta}(x-0)\right]\mathrm{d}x \\ &= \overline{\boldsymbol{\lambda}}_{\mathrm{g}}^{\mathrm{T}}\int_0^\ell \left[E\frac{h}{2}\begin{bmatrix}\frac{6}{\ell^2}+12\frac{x}{\ell^3} \\ \frac{-4}{\ell}+6\frac{x}{\ell^2} \\ \frac{6}{\ell^2}-12\frac{x}{\ell^3} \\ \frac{-2}{\ell}+6\frac{x}{\ell^2}\end{bmatrix}\widehat{\delta}(x-0)\right]\mathrm{d}x = \overline{\boldsymbol{\lambda}}_{\mathrm{g}}^{\mathrm{T}}\begin{bmatrix}\frac{3Eh}{\ell^2} \\ \frac{-2Eh}{\ell} \\ \frac{3Eh}{\ell^2} \\ \frac{-Eh}{\ell}\end{bmatrix} = \overline{\boldsymbol{\lambda}}_{\mathrm{g}}^{\mathrm{T}}\mathbf{F}_{\mathrm{adj_g}}\end{aligned} \tag{18.186}$$

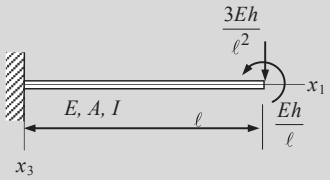
By imposing the boundary conditions and removing the virtual adjoint displacement $\overline{\boldsymbol{\lambda}}$ from both sides, we have $\mathbf{K}\boldsymbol{\lambda} = \mathbf{F}_{\mathrm{adj}}$. That is, the adjoint load vector $\mathbf{F}_{\mathrm{adj}}$ is

$$\mathbf{F}_{\mathrm{adj}} = \begin{bmatrix}\frac{3Eh}{\ell^2} \\ \frac{-Eh}{\ell}\end{bmatrix} \tag{18.187}$$

Continued

EXAMPLE 18.14—cont'd

which is a point load $\frac{3Eh}{\ell^2}$ applied at the tip of the beam in the downward direction and a point moment $\frac{Eh}{\ell}$ applied at the tip of the beam in the negative direction (counterclockwise), as shown in the figure below.



Solve the adjoint response by using the finite element equation, similar to Eq. 18.148:

$$\boldsymbol{\lambda} = \mathbf{K}^{-1}\mathbf{F}_{\text{adj}} = \frac{\ell}{12EI}\begin{bmatrix} 4\ell^2 & 6\ell \\ 6\ell & 12 \end{bmatrix}\begin{bmatrix} \dfrac{3Eh}{\ell^2} \\ -\dfrac{Eh}{\ell} \end{bmatrix} = \begin{bmatrix} \dfrac{h\ell}{2I} \\ \dfrac{h}{2I} \end{bmatrix} \tag{18.188}$$

Hence

$$\lambda_{3,11} = \begin{bmatrix} N_{3,11} & N_{4,11} \end{bmatrix}\begin{bmatrix} \dfrac{h\ell}{2I} \\ \dfrac{h}{2I} \end{bmatrix} = \begin{bmatrix} \dfrac{6}{\ell^2} - \dfrac{12x}{\ell^3} & \dfrac{-2}{\ell} + \dfrac{6x}{\ell^2} \end{bmatrix}\begin{bmatrix} \dfrac{h\ell}{2I} \\ \dfrac{h}{2I} \end{bmatrix} = \frac{h}{I\ell}\left(2 - \frac{3x}{\ell}\right) \tag{18.189}$$

By bringing the original responses $z_{3,11}$ and adjoint responses $\lambda_{3,11}$ to the sensitivity equation (Eq. 18.178), we have

$$\begin{aligned} \psi' &= \int_0^\ell g_{,b}\delta b \mathrm{d}x - d'_{\delta b}(\mathbf{z}, \boldsymbol{\lambda}) + \ell'_{\delta b}(\boldsymbol{\lambda}) \\ &= \int_0^\ell \left[E\frac{\delta h}{2} z_{3,11}(x)\widehat{\delta}(x-0)\right]\mathrm{d}x - \int_0^\ell EI' z_{3,11}\lambda_{3,11}\mathrm{d}x + 0 \\ &= E\frac{\delta h}{2} z_{3,11}(0) - \int_0^\ell E\frac{3I}{h}\left[\frac{q\ell}{12EI}(5\ell - 6x)\right]\frac{h}{I\ell}\left(2 - \frac{3x}{\ell}\right)\mathrm{d}x\delta h \\ &= \frac{5q\ell^2}{24I}\delta h - \frac{5q\ell^2}{8I}\delta h = -\frac{5q\ell^2}{12I}\delta h \end{aligned} \tag{18.190}$$

18.4.3.6 Adjoint Variable Method of the Continuum-Analytical Approach

We define a compliance performance measure and derive the sensitivity expression for the measure using the adjoint variable method in the continuum form. To simplify the formulation, we assume both spatial variable x and state variable z are scalar quantities. We then use the analytical solution of the cantilever beam example again to illustrate a few details of the continuum-analytical approach.

A compliance performance measure of a structure is defined as

$$\psi = \int_{\Omega} f(x) z \mathrm{d}\Omega \tag{18.191}$$

where $f(x)$ is the external load per length (or per area and per volume for two- and three-dimensional problems, respectively) applied to the structure. Compliance is a global measure; a smaller compliance implies that the structure is more rigid. As shown in Eq. 18.191 for a less compliant structure, a smaller displacement is expected for a given external load $f(x)$.

If we assume a uniformly distributed load applied to the cantilever beam, as shown in Figure 18.11(a), f(x) is a uniformly distributed load per length; that is, $f(x) = q$. If we also include the self-weight of the beam, the force $f(x)$ per length can be written as

$$f(x) = q + \gamma A \tag{18.192}$$

where γ is the weight density. Note that for the cantilever beam shown in Figure 18.11(a), the displacement is z_3. Hence, Eq. 18.191 becomes

$$\psi = \int_0^{\ell} (q + \gamma A) z_3 \mathrm{d}x \tag{18.193}$$

Taking the variation of the performance measure defined in Eq. 18.193, we have

$$\psi' = \int_0^{\ell} (\gamma A)_{,b} \delta b z_3 \mathrm{d}x + \int_0^{\ell} (q + \gamma A) z_3' \mathrm{d}x \tag{18.194}$$

If the height h is the design variable, then $(\gamma A)_{,h}\delta h = \gamma (wh)_{,h}\delta h = \gamma w \delta h$. We create an adjoint structure as

$$a_b(\boldsymbol{\lambda}, \overline{\boldsymbol{\lambda}}) = \int_0^{\ell} (q + \gamma A) \overline{\lambda}_3 \mathrm{d}x \tag{18.195}$$

in which the right-hand side is identical to the load linear form $\ell_b(\overline{z}) = \int_0^{\ell} (q+\gamma A)\overline{z}_3 \mathrm{d}x$. Hence, solving Eq. 18.195 for λ_3 is identical to solving the original structure for z_3, implying that the adjoint response is identical to the original structural response; that is, $\lambda_3 = z_3$. This is called self-adjoint and is only true for a compliance performance measure. The sensitivity equation becomes

$$\psi' = \int_0^{\ell} \left[(\gamma A)_{,b} \delta b \right] z_3 \mathrm{d}x - a'_{\delta b}(z, z) + \ell'_{\delta b}(z) \tag{18.196}$$

EXAMPLE 18.15

The displacement of the cantilever beam due to the distributed load q and self-weight γA is

$$z_3^{\text{Exact}}(x) = \frac{q+\gamma A}{24EI}\left(6x^2\ell^2 - 4x^3\ell + x^4\right) \tag{18.197}$$

Calculate the sensitivity of the compliance performance measure for the area design variable A and modulus design variable E using the continuum-analytical approach.

Solution

Twice taking the derivatives of Eq. 18.197 with respect to x, we have

$$z_{3,11}(x) = \frac{q+\gamma A}{2EI}\left(\ell^2 - 2x\ell + x^2\right) = \frac{q+\gamma A}{2EI}(\ell - x)^2 \tag{18.198}$$

From Eq. 18.196, we have

$$\begin{aligned}
\psi' &= \int_0^\ell (\gamma A)_{,b}\delta b z_3 \mathrm{d}x - a'_{\delta b}(z,z) + \ell'_{\delta b}(z) \\
&= \int_0^\ell (\gamma w) z_3 \mathrm{d}x \delta h - \int_0^\ell \left(E\frac{3I}{h}\delta h + I\delta E\right) z_{3,11}^2 \mathrm{d}x \delta h + \int_0^\ell (\gamma w) z_3 \mathrm{d}x \delta h \\
&= \left\{2\int_0^\ell (\gamma w) z_3 \mathrm{d}x - \int_0^\ell E\frac{3I}{h} z_{3,11}^2 \mathrm{d}x\right\}\delta h + \left\{-\int_0^\ell I z_{3,11}^2 \mathrm{d}x\right\}\delta E \\
&= \frac{(q+\gamma A)\ell^5}{20EI}\left(2\gamma w - 3\frac{q+\gamma A}{h}\right)\delta h - \frac{(q+\gamma A)^2\ell^5}{20E^2 I}\delta E
\end{aligned} \tag{18.199}$$

The sensitivity coefficient of design variable E is negative, implying that increasing the modulus E decreases the compliance of the structure (the cantilever beam becomes more rigid). On the other hand, the sign of the sensitivity coefficient of design variable h depends on the changes in self-weight and change in the moment of inertia due to the change in height h. Increasing h increases the compliance due to the increment in self-weight. Increasing h makes the beam more rigid, hence decreasing the compliance of the structure.

18.4.4 NUMERICAL IMPLEMENTATION

As mentioned in Section 18.3.4, sensitivity equations derived from the continuum-discrete approach, either direct differentiation method or adjoint variable method, can be implemented external to commercial FEA codes that are employed for structural analysis.

Numerical computation of Eq. 18.131 (direct differentiation method) or Eq. 18.178 (adjoint variable method) requires knowledge of the original structural response, $\mathbf{z}$, and/or the adjoint structural response, $\boldsymbol{\lambda}$. The solution $\mathbf{z}$ of Eq. 18.106 is obtained by structural finite element analysis. Using the adjoint variable method, the solution $\boldsymbol{\lambda}$ of the adjoint equation (Eq. 18.174) can be obtained by restarting the finite element analysis code that was used for the analysis model but with additional loading vectors that are defined by the right-hand side of Eq. 18.174. Notice that Eq. 18.174 has to be solved for each displacement or stress performance measure that has a corresponding adjoint load. For other performance measures, such as compliance, natural frequency, buckling load, volume, and mass performance measures, no adjoint structural analysis is necessary.

Using the direct differentiation method, fictitious loads must be calculated by carrying out numerical integration of the terms on the right-hand side of Eq. 18.131 over the entire structural domain. Note that Eq. 18.131 has to be solved for each design variable per load case.

For implementation with an existing FEA code, there are four software programs that need to be developed: (1) an interface program that retrieves results $\mathbf{z}$ and/or $\boldsymbol{\lambda}$ from the database of the FEA code, (2) a program that generates input data file of the FEA code for reanalysis with either adjoint load or fictitious load, (3) a program that performs numerical integration to evaluate the terms on the right-hand side of Eq. 18.131 and/or Eq. 18.174, and (4) a script that integrates programs to carry out batch mode computation. Note that computation for the fictitious load or sensitivity coefficients must be carried out numerically by using, for example, Gauss integration (Atkinson, 1989) since the FEA results are given in numerical data instead of notations as seen in the examples of this section.

18.5 SHAPE SENSITIVITY ANALYSIS*

Shape sensitivity analysis characterizes the influence of structural geometric shape change to its performance. For frame structures consisting of bars or beams, a change in dimension variables causes changes in the length and orientation angle of individual bar or beam members in the structure, as shown in Figure 18.2(b). Length change is referred to as domain shape change, and angle change is referred to as configuration change. For 2-D planar and 3-D solid structures, shape sensitivity analysis only involves domain shape change. In this section, we discuss domain shape sensitivity analysis. For those who are interested in learning more about configuration design for frame structures, Choi and Kim (2006a) offers excellent details. To avoid the complex underlying mathematical formulation involved in shape sensitivity analysis theory, our discussion is focused more on the practical aspects. However, because the basic concepts cannot be introduced without discussing the theory, we use a simple cantilever beam example to minimize complex mathematical formulations.

18.5.1 DOMAIN SHAPE SENSITIVITY ANALYSIS

In the shape design of 2-D planar or 3-D solid structures, the component under consideration is typically a continuum, and its shape is defined by geometric boundaries. Usually, a portion of the geometric boundary is designated as the design boundary for shape sensitivity analysis (and optimization). For a 2-D planar structure, the design boundary is a curve (or composite curve). For 3-D problems, the design boundary is a surface (or composite surface). In order to carry out shape sensitivity analysis, the design boundary of the structural component must be parameterized in a mathematical form that is compatible with computer-aided design (CAD) so that follow-up engineering assignments involved in product design, such as machining or manufacturing process planning, can be readily carried out. The design boundary can be represented by freeform curves or surfaces (e.g., a Bézier curve or a B-spline surface) or by regular geometric curves and surfaces (e.g., an elliptical arc, a cylindrical surface) commonly found in CAD. Shape parameterization of freeform surfaces as well as regular CAD geometric features, will be discussed in Section 18.5.3.

The second key issue in shape optimization is design velocity field computation (Choi and Chang, 1994). The design velocity field characterizes the movement of material points of the structural domain

due to changes at the boundary. In practice, the design velocity field is also used to update finite element mesh during shape optimization. Updating mesh using design velocity retains the topology of the finite element mesh throughout the design iterations; therefore, the consistency of performance measures evaluated using FEA throughout design iterations is ensured. This mesh update is especially critical for tracing local performance measures, such as displacements or stresses, during design iterations. The design velocity field must comply with the geometric shape of the boundary; that is, all finite element nodes at the design boundary must stay on the boundary when the design is varied. It is critical that one single geometric representation at the design boundary supports structural analysis, sensitivity analysis, and follow-on engineering tasks, such as machining. Design velocity field computation and related issues are briefly explained in Section 18.5.4.

After the design boundary is parameterized and design velocity field is calculated, shape sensitivity analysis can be carried out. The easiest approach for shape sensitivity analysis is probably the overall finite difference. The semi-analytical method is more efficient than the overall finite difference. Like sizing sensitivity analysis, these methods require an adequate design perturbation size for accurate sensitivity coefficients. On the other hand, the continuum approach is efficient and does not require any design perturbations. We discuss the finite difference method and continuum approach in Sections 18.5.5 and 18.5.7, respectively. Before getting into these subjects, in the next subsection we use a simple cantilever beam to illustrate some of the basic but essential concepts involved in shape sensitivity analysis.

18.5.2 A SIMPLE CANTILEVER BEAM EXAMPLE

Shape design involves altering the geometric shape of the structure for improved or optimal performance. For a simple case, such as the cantilever beam discussed in Section 18.4.3 and shown in Figure 18.13(a) again, the displacement obtained from FEA is

$$z_3(x) = \frac{q\ell}{24EI}\left(5x^2\ell - 2x^3\right) \tag{18.200}$$

At the tip, the displacement is

$$z_3^{\ell} = z_3(\ell) = \frac{q\ell}{24EI}(5\ell^3 - 2\ell^3) = \frac{q\ell^4}{8EI} \tag{18.201}$$

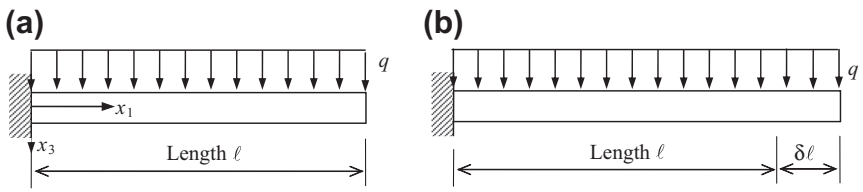


FIGURE 18.13

Cantilever beam example: (a) current design and (b) perturbed design with length $\ell + \delta\ell$, in which the overall force is increased from $q\ell$ to $q(\ell + \delta\ell)$.

For a cantilever beam, the only shape design variable is its length. The easiest way of calculating shape sensitivity coefficient for the displacement is by taking the derivative of z_3^ℓ with respect to length ℓ:

$$\frac{\partial z_3^\ell}{\partial \ell} = \frac{\partial}{\partial \ell}\frac{q\ell^4}{8EI} = \frac{q\ell^3}{2EI} \tag{18.202}$$

which is nothing but the analytical derivative method discussed in Section 18.3.1. Because the sign of the derivative in Eq. 18.202 is positive, it is implied that increasing the length of beam increases its displacement at the tip. Such an increment is due to three factors: the increment in beam length, the increment in the overall load from $q\ell$ to $q(\ell+\delta\ell)$, and the movement of the displacement measurement point that stays at the tip of the beam (i.e., moving with the length change).

There are different approaches to calculating the shape sensitivity of the beam. The analytical approach is shown in Eq. 18.202. Another simple way for calculating the sensitivity coefficient is the discrete-analytical approach. Similar to Eq. 18.40 formulated for sizing design variables, for the length design variable, we have

$$\mathbf{K}\frac{\partial \mathbf{Z}}{\partial \ell} = \frac{\partial \mathbf{F}}{\partial \ell} - \frac{\partial \mathbf{K}}{\partial \ell}\mathbf{Z} \tag{18.203}$$

in which

$$\frac{\partial \mathbf{K}}{\partial \ell} = \frac{\partial}{\partial \ell}\left\{ EI\begin{bmatrix} \frac{12}{\ell^3} & -\frac{6}{\ell^2} \\ -\frac{6}{\ell^2} & \frac{4}{\ell} \end{bmatrix}\right\} = EI\begin{bmatrix} -\frac{36}{\ell^4} & \frac{12}{\ell^3} \\ \frac{12}{\ell^3} & -\frac{4}{\ell^2} \end{bmatrix} \tag{18.204}$$

and

$$\frac{\partial \mathbf{F}}{\partial \ell} = \frac{\partial}{\partial \ell}\begin{bmatrix} \frac{1}{2}q\ell \\ -\frac{1}{12}q\ell^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}q \\ -\frac{1}{6}q\ell \end{bmatrix} \tag{18.205}$$

Collecting the terms on the right-hand side of Eq. 18.203, we have the fictitious load vector:

$$\mathbf{F}_{\text{fic}} = \frac{\partial \mathbf{F}}{\partial \ell} - \frac{\partial \mathbf{K}}{\partial \ell}\mathbf{Z} = \begin{bmatrix} \frac{1}{2}q \\ -\frac{1}{6}q\ell \end{bmatrix} - EI\begin{bmatrix} -\frac{36}{\ell^4} & \frac{12}{\ell^3} \\ \frac{12}{\ell^3} & -\frac{4}{\ell^2} \end{bmatrix}\begin{bmatrix} \frac{q\ell^4}{8EI} \\ \frac{q\ell^3}{6EI} \end{bmatrix} = \begin{bmatrix} 3q \\ -q\ell \end{bmatrix} \tag{18.206}$$

which is a point load $3q$ acting at the tip in the downward direction, and a point moment $q\ell$ at the tip acting in a counterclockwise direction. Solving for Eq. 18.203, we have

$$\begin{bmatrix} \frac{\partial z_3^\ell}{\partial \ell} \\ \frac{\partial \theta}{\partial \ell} \end{bmatrix} = \mathbf{K}^{-1}\mathbf{F}_{\text{fic}} = \frac{\ell}{12EI}\begin{bmatrix} 4\ell^2 & 6\ell \\ 6\ell & 12 \end{bmatrix}\begin{bmatrix} 3q \\ -q\ell \end{bmatrix} = \begin{bmatrix} \frac{q\ell^3}{2EI} \\ \frac{q\ell^2}{2EI} \end{bmatrix} \tag{18.207}$$

in which $\frac{\partial z_3^\ell}{\partial \ell} = \frac{q\ell^3}{2EI}$, which is same as that of Eq. 18.202.

Now, let us try a slightly different way to calculate the displacement sensitivity. Instead of taking the derivative of the displacement at the tip z_3^ℓ with respect to length ℓ, we take the derivative of the function $z_3(x)$ with respect to length ℓ, then evaluate the derivative at the tip $x = \ell$. Taking the derivative of $z_3(x)$ with respect to length ℓ, we have

$$\frac{\partial z_3(x)}{\partial \ell} = \frac{\partial}{\partial \ell}\left[\frac{q}{24EI}\left(5x^2\ell^2 - 2x^3\ell\right)\right] = \frac{qx^2}{12EI}(5\ell - x) \tag{18.208}$$

At the tip of the beam, the displacement sensitivity is

$$\frac{\partial z_3}{\partial \ell} = \left.\frac{\partial z_3(x)}{\partial \ell}\right|_{x=\ell} = \frac{q\ell^3}{3EI} \tag{18.209}$$

Compared Eq. 18.209 to 18.202, the two sensitivity coefficients $\frac{\partial z_3}{\partial \ell}$ and $\frac{\partial z_3^\ell}{\partial \ell}$ are not identical. Why? What do these two sensitivity coefficients mean? To answer this question, let us first rewrite Eq. 18.202 into a finite difference form:

$$\begin{aligned}\frac{\partial z_3^\ell}{\partial \ell} \approx \frac{\Delta z_3^\ell}{\Delta \ell} &= \frac{z_3^\ell(\ell + \delta\ell) - z_3^\ell(\ell)}{\delta\ell} = \frac{\frac{q(\ell+\delta\ell)^4}{8EI} - \frac{q\ell^4}{8EI}}{\delta\ell} \\ &= \frac{q}{8EI}\,\frac{4\ell^3\delta\ell + 6\ell^2\delta\ell^2 + 4\ell\delta\ell^3 + \delta\ell^4}{\delta\ell} = \frac{q\ell^3}{2EI} + \frac{q}{8EI}\left(6\ell^2\delta\ell + 4\ell\delta\ell^2 + \delta\ell^3\right)\end{aligned} \tag{18.210}$$

in which $z_3^\ell(\ell + \delta\ell) = \frac{q(\ell+\delta\ell)^4}{8EI}$ represents the displacement at the tip of the beam after a design change; that is, at $x = \ell + \delta\ell$, as illustrated in Figure 18.14. Certainly, only the first term survives in Eq. 18.210 when we let the design change $\delta\ell$ approach zero.

Now, we rewrite Eq. 18.208 in the finite difference form:

$$\begin{aligned}\frac{\partial z_3(x)}{\partial \ell} \approx \frac{\Delta z_3(x)}{\Delta \ell} &= \frac{z_3(x;\ell + \delta\ell) - z_3(x;\ell)}{\delta\ell} \\ &= \frac{\frac{q}{24EI}\left(5x^2(\ell+\delta\ell)^2 - 2x^3(\ell+\delta\ell)\right) - \frac{q}{24EI}\left(5x^2\ell^2 - 2x^3\ell\right)}{\delta\ell} \\ &= \frac{q}{24EI}\,\frac{2x^2(5x-\ell)\delta\ell + 5x^2\delta\ell^2}{\delta\ell} = \frac{qx^2}{12EI}(5x-\ell) + \frac{q}{24EI}\left(5x^2\delta\ell\right)\end{aligned} \tag{18.211}$$

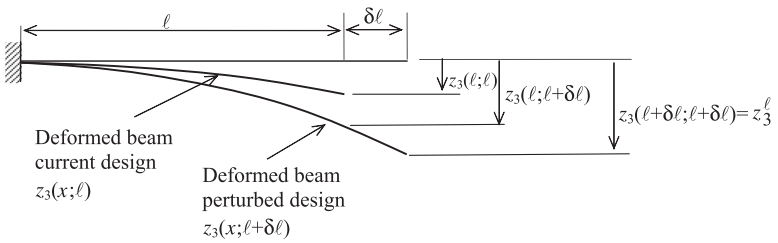


FIGURE 18.14

Deformed cantilever beam in the current and perturbed designs.

in which $z_3(x;\ell+\delta\ell)$ represents displacement function z_3 at the perturbed design:

$$z_3(x;\ell+\delta\ell)=\frac{q}{24EI}\Big(5x^2(\ell+\delta\ell)^2-2x^3(\ell+\delta\ell)\Big) \tag{18.212}$$

If we evaluate Eq. 18.212 at the tip of the perturbed design—that is, at $x=\ell+\delta\ell$—then

$$z_3(\ell+\delta\ell;\ell+\delta\ell)=\frac{q}{24EI}\Big(5(\ell+\delta\ell)^4-2(\ell+\delta\ell)^4\Big)=\frac{q}{8EI}(\ell+\delta\ell)^4 \tag{18.213}$$

which is $z_3^{\ell}(\ell+\delta\ell)$. However, in Eq. 18.208, we assume $x=\ell$; that is, the measurement point is stationary, not moving with the design—hence, the result of Eq. 18.209.

As illustrated above, different shape sensitivity coefficients are obtained depending on whether the measurement point is stationary.

Another important observation to point out is the measurement point inside the beam—that is, $0<x<\ell$, for example, at the midpoint $x=\ell/2$. How does this measurement point move with design? Is it moving half the length change $\delta\ell/2$, assuming the interior material moves proportionally with design, as shown in Figure 18.15(a)? Or, can the midpoint movement be determined in another way, such as by a quadratic function, as shown in Figure 18.15(b)?

In fact, in shape design, the material point movement is determined by the design velocity field, which plays an important role in shape sensitivity analysis (and optimization). The sensitivity we calculated predicts the performance measure of the material point whose location is determined by the velocity field employed in the calculation. The material point location at the next design iteration is also determined by the design velocity field. These key concepts illustrated using the beam example will be extended to more complex and general problems in the following discussion.

Although shape sensitivity analysis can be carried out easily for this cantilever beam example, the application of shape design for one-dimensional structures is limited. As mentioned at the beginning of the section, for a frame structure, when its geometric shape changes, individual components (beams or bars) experience not only length changes but also changes in orientation angles—the so-called configuration changes. In this section, we focus on domain shape design. Domain shape designs are often seen in two-dimensional planar or three-dimensional solid structural components.

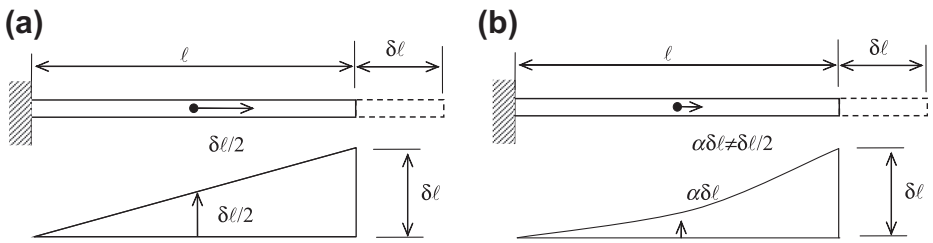


FIGURE 18.15

Midpoint movement with design change $\delta\ell$: (a) $\delta\ell/2$: linear velocity and (b) $\alpha\delta\ell\neq\delta\ell/2$, $\alpha\neq 0$: other than linear velocity.

18.5.3 SHAPE DESIGN PARAMETERIZATION

Shape design variables govern the geometric shape of the structural boundary, usually represented by parametric curves and surfaces for 2-D and 3-D applications, respectively. It is important to select a parameterization scheme that properly reflects the design intent. If you are not familiar with parametric curves and surfaces, you are encouraged to review Chapter 2 of this book before reading this subsection. The following explains parameterization of a few selected geometric entities for 2-D and 3-D structures. These geometric entities are supported in CAD in addition to widely accepted geometric modeling tools, such as MSC/PATRAN (www.mscsoftware.com/product/patran) and HyperMesh (www.altairhyperworks.com).

18.5.3.1 2-D Planar Structures

For 2-D structures, the design boundaries are planar curves. To stay focused, we assume the design boundary is represented by a cubic curve. There are eight degrees of freedom for a planar parametric cubic curve:

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}_{1\times 4} \begin{bmatrix} a_{3x} & a_{3y} \\ a_{2x} & a_{2y} \\ a_{1x} & a_{1y} \\ a_{0x} & a_{0y} \end{bmatrix}_{4\times 2} = \mathbf{U}_{1\times 4}\mathbf{A}_{4\times 2}, \quad u \in [0,1] \tag{18.214}$$

where u is the parametric coordinate of the curve. The 4×2 matrix $\mathbf{A}$ contains algebraic coefficients $\mathbf{a}_i$ that determine the geometric shape of the curves. Equation (18.214) is called the algebraic format. In practice, boundary curves are not parameterized using these algebraic coefficients because the change of coefficients $\mathbf{a}_i$ to curve geometry is not clear.

There are several widely accepted formats by which the curves can be better parameterized, such as the spline curve, Hermit cubic curve, Bézier curve, and B-spline curve (see Chapter 2), where the influences of the parameter change to the geometric shape of the curve are obvious. Mathematically, these curves are represented as a set of basis functions $\mathbf{B}(u)$ and geometric control matrix $\mathbf{G}$ that does not depend on the parametric coordinate u:

$$\mathbf{P}(u) = \mathbf{U}(u)\mathbf{A} = \mathbf{U}(u)\mathbf{N}\mathbf{G} = \mathbf{B}(u)\mathbf{G} \tag{18.215}$$

Here, the basic functions can be written as a multiplication of the vector $\mathbf{U}$ and a matrix $\mathbf{N}$ that is a constant 4×4 matrix determined by the respective curve format. For a planar cubic curve, $\mathbf{G}$ is a 4×2 matrix that controls the geometric shape of the curve. The entries of the matrix $\mathbf{G}$ are selected as shape design variables.

We assume a cubic Bézier curve to illustrate further. The cubic Bézier curve shown in Figure 18.16(a) is determined by the position of its four control points $\mathbf{G}$, which form a control polygon. Mathematically, a Bézier curve can be written in a form like that of Eq. 18.215:

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} = \mathbf{U}\mathbf{N}^{\mathrm{B}}\mathbf{G}^{\mathrm{B}} = \mathbf{B}^{\mathrm{B}}\mathbf{G}^{\mathrm{B}} \tag{18.216}$$

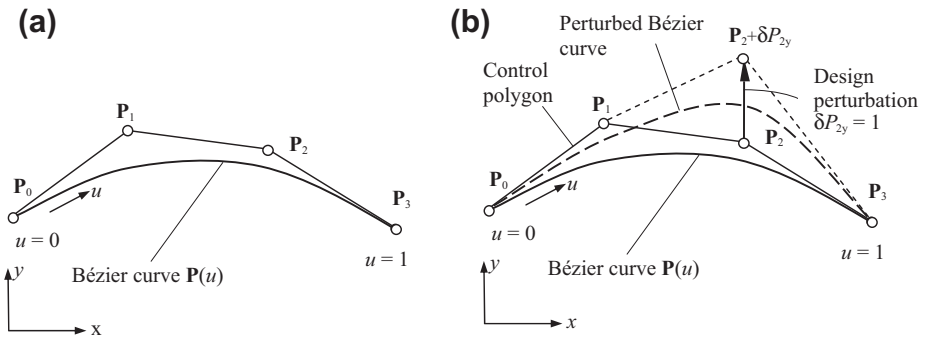


FIGURE 18.16

(a) A cubic Bézier curve parameterized by four control points that form a control polygon. (b) The perturbed curve by moving control point $\mathbf{P}_2$ in y-direction.

where

$$\mathbf{G}^{\mathrm{B}} = \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} = \begin{bmatrix} P_{0x} & P_{0y} \\ P_{1x} & P_{1y} \\ P_{2x} & P_{2y} \\ P_{3x} & P_{3y} \end{bmatrix}_{4\times 2} \tag{18.217}$$

and

$$\mathbf{N}^{\mathrm{B}} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}_{4\times 4} \tag{18.218}$$

which is symmetric. Hence, the basis functions, called Bernstein polynomials, can be obtained as

$$\mathbf{U}\mathbf{N}^{\mathrm{B}} = \left[-u^3 + 3u^2 - u + 1, u^3 - 6u^2 + 3u, -u^3 + 3u^2, u^3 \right]_{1\times 4} \tag{18.219}$$

Movement of any of the four control points in the x- or y-direction can be chosen as a shape design variable. Any change in the position of the control points will result in a change of the geometric shape of the curve through the basis functions. For example, if the control point $\mathbf{P}_2$ moves vertically up to a new position by δP_{2y}, as shown in Figure 18.16(b), the change in curve can be obtained as

$$\delta\mathbf{P}(u) = \mathbf{U}\mathbf{N}^{\mathrm{B}}\delta\mathbf{G}^{\mathrm{B}} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \delta P_{2y} \\ 0 & 0 \end{bmatrix} \tag{18.220}$$

The perturbed curve $\mathbf{P}(u;\,\delta\mathbf{G})$ can be obtained as

$$\mathbf{P}(u;\,\delta\mathbf{G}) = \mathbf{P}(u) + \delta\mathbf{P}(u) \tag{18.221}$$

EXAMPLE 18.16

Given four control points—$\mathbf{P}_0 = [0, 0]$, $\mathbf{P}_1 = [1, 3]$, $\mathbf{P}_2 = [2, -2]$, and $\mathbf{P}_3 = [3, 0]$—compute the parametric equation of the Bézier curve formed by them. If the control point $\mathbf{P}_2$ is moved upward by two units—that is, $\delta P_{2y} = 2$—calculate the change in curve $\delta\mathbf{P}(u)$, and calculate the parametric equation of the perturbed Bézier curve.

Solution

Using Eq. 18.216, we have

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 3 \\ 2 & -2 \\ 3 & 0 \end{bmatrix} = \begin{bmatrix} 3u, & 15u^3 - 24u^2 + 9u \end{bmatrix} \tag{18.222}$$

The change in the curve due to $\delta P_{2y} = 2$ can be obtained using Eq. 18.220 as

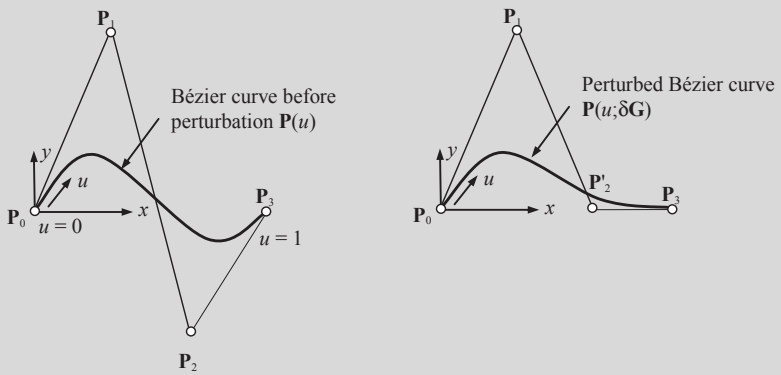
$$\delta\mathbf{P}(u) = \mathbf{U}\mathbf{N}^{\mathrm{B}}\delta\mathbf{P}^{\mathrm{B}} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 2 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 0 & -6 \\ 0 & 6 \\ 0 & 2 \\ 0 & 0 \end{bmatrix} \tag{18.223}$$

$$= \begin{bmatrix} 0, -6u^3 + 6u^2 \end{bmatrix}$$

The perturbed curve $\mathbf{P}(u;\,\delta\mathbf{G})$ can be obtained using Eq. 18.221 as

$$\mathbf{P}(u;\,\delta\mathbf{G}) = \mathbf{P}(u) + \delta\mathbf{P}(u) = \begin{bmatrix} 3u, 9u^3 - 18u^2 + 9u \end{bmatrix} \tag{18.224}$$

The Bézier curves before and after perturbation are shown below:



18.5.3.2 3-D Solid Structures—Freeform Surfaces

For a 3-D solid structure, its design boundary can be modeled as a spatial parametric surface or CAD-generated surface if a CAD solid model is considered. A spatial parametric surface, referred to as freeform parametric surfaces in this subsection, such as a Coons patch, ruled surface, Bézier surface, and B-spline (or NURB) surface (Mortenson, 2006), are the popular choices for shape parameterization. A parametric surface can be represented by a parametric vector equation $\mathbf{S}$ and a parametric area $A \in R^2$, usually $A = [0,1] \times [0,1]$, such that the surface consists of the set of points $\{\mathbf{S}(u,w) | (u,w) \in A\}$, as illustrated in Figure 18.17. Note that $\mathbf{S}$ is sometimes called the evaluation function and A is called the domain of evaluation.

We assume that the design boundary is represented by a bicubic parametric surface, cubic in both its parametric coordinates u and w. In general, there are 48 degrees of freedom for a parametric bicubic surface. The mathematical expressions in algebraic format for a bicubic parametric surface are as follows:

$$\begin{aligned} S_x(u,w) &= \sum_{i,j=0}^{3} a_{ijx} u^i w^j = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_{33x} & a_{32x} & a_{31x} & a_{30x} \\ a_{23x} & a_{22x} & a_{21x} & a_{20x} \\ a_{13x} & a_{12x} & a_{11x} & a_{10x} \\ a_{03x} & a_{02x} & a_{01x} & a_{00x} \end{bmatrix}_{4\times 4} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix} \\ &= \mathbf{U}\mathbf{A}_x\mathbf{W}^{\mathrm{T}}, \quad (u,w) \in [0,1] \times [0,1] \end{aligned} \tag{18.225a}$$

$$\begin{aligned} S_y(u,w) &= \sum_{i,j=0}^{3} a_{ijy} u^i w^j = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_{33y} & a_{32y} & a_{31y} & a_{30y} \\ a_{23y} & a_{22y} & a_{21y} & a_{20y} \\ a_{13y} & a_{12y} & a_{11y} & a_{10y} \\ a_{03y} & a_{02y} & a_{01y} & a_{00y} \end{bmatrix}_{4\times 4} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix} \\ &= \mathbf{U}\mathbf{A}_y\mathbf{W}^{\mathrm{T}}, \quad (u,w) \in [0,1] \times [0,1] \end{aligned} \tag{18.225b}$$

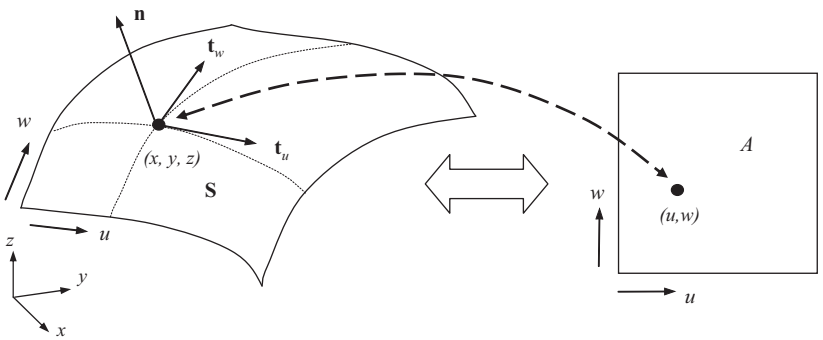


FIGURE 18.17

Parametric surface **S** and its parametric area *A*.

and

$$S_z(u,w) = \sum_{i,j=0}^{3} a_{ijz} u^i w^j = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_{33z} & a_{32z} & a_{31z} & a_{30z} \\ a_{23z} & a_{22z} & a_{21z} & a_{20z} \\ a_{13z} & a_{12z} & a_{11z} & a_{10z} \\ a_{03z} & a_{02z} & a_{01z} & a_{00z} \end{bmatrix}_{4\times4} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix}$$
$$= \mathbf{U}\mathbf{A}_z\mathbf{W}^T, \quad (u,w) \in [0,1] \times [0,1] \tag{18.225c}$$

For convenience, we have combined the above equations as

$$S(u,w) = \sum_{i,j=0}^{3} \mathbf{a}_{ij} u^i w^j = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_{33} & \mathbf{a}_{32} & \mathbf{a}_{31} & \mathbf{a}_{30} \\ \mathbf{a}_{23} & \mathbf{a}_{22} & \mathbf{a}_{21} & \mathbf{a}_{20} \\ \mathbf{a}_{13} & \mathbf{a}_{12} & \mathbf{a}_{11} & \mathbf{a}_{10} \\ \mathbf{a}_{03} & \mathbf{a}_{02} & \mathbf{a}_{01} & \mathbf{a}_{00} \end{bmatrix}_{4\times4\times3} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix}$$
$$= \mathbf{U}\mathbf{A}\mathbf{W}^{\mathrm{T}}, \quad (u,w) \in [0,1] \times [0,1] \tag{18.226}$$

where $\mathbf{S}(u,w) = [S_x, S_y, S_z]$. Note that in Eq. 18.226 $\mathbf{a}_{ij} = [\mathbf{a}_{ijx}, \mathbf{a}_{ijy}, \mathbf{a}_{ijz}]$, 4×4 matrix each, are the algebraic coefficients of the surface. Similar to the parametric curves, in practice, the algebraic format of the parametric surface is not suitable for shape parameterization. Two surfaces that are commonly employed to parameterize design surfaces are Bézier and B-spline (or NURB) surfaces. We use a bicubic Bézier surface to illustrate further.

A bicubic Bézier surface, shown in Figure 18.18, is determined by the position of its 16 control points $\mathbf{G}$, which form a control polyhedron. Mathematically, a Bézier surface can be written in a form like that of Eq. 18.226 as

$$\begin{aligned} \mathbf{S}(u,w) &= \mathbf{U}\mathbf{N}^{\mathrm{B}}\mathbf{G}\mathbf{N}^{\mathbf{B}^{\mathrm{T}}}\mathbf{W}^{\mathrm{T}} \\ &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \\ &\quad \times \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{10} & \mathbf{p}_{20} & \mathbf{p}_{30} \\ \mathbf{p}_{01} & \mathbf{p}_{11} & \mathbf{p}_{21} & \mathbf{p}_{31} \\ \mathbf{p}_{02} & \mathbf{p}_{12} & \mathbf{p}_{22} & \mathbf{p}_{32} \\ \mathbf{p}_{03} & \mathbf{p}_{13} & \mathbf{p}_{23} & \mathbf{p}_{33} \end{bmatrix}_{4\times4\times3} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \\ &\quad \times \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix}, \quad (u,w) \in [0,1] \times [0,1] \end{aligned} \tag{18.227}$$

where $\mathbf{P}_{ij}$ is the control point at the ith row and jth column of the control point matrix $\mathbf{G}$ of $4 \times 4 \times 3$, u and w are the parametric coordinates of the surface, and $\mathbf{U}\mathbf{N}^{\mathrm{B}}$ and $\mathbf{N}^{\mathrm{B}}\mathbf{W}^{\mathrm{T}}$ give cubic Bernstein polynomials in u and w, respectively.

Similar to the Bézier curve, movement of any of the 16 control points in the x-, y-, or z-direction can be chosen as the shape design variable. Any change in the position of the control points

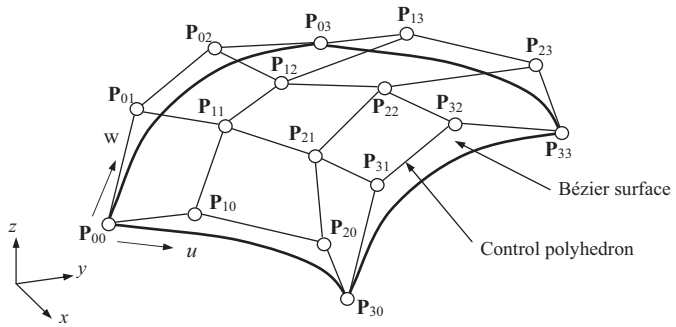


FIGURE 18.18

A bicubic Bézier surface determined by a control polyhedron of 4 × 4 control points.

will result in the change of the geometric shape of the surface. For example, if control point $\mathbf{P}_{22}$ moves vertically up to a new position in the y-direction by δP_{22y}, the change in the surface can be obtained as

$$\begin{aligned}\delta S_y(u,w) &= \mathbf{U}\mathbf{N}^{\mathrm{B}}\delta\mathbf{G}_y\mathbf{N}^{\mathrm{B}}\mathbf{W}^{\mathrm{T}} \\ &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \delta P_{22y} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \\ &\quad \times \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix}, \quad (u,w) \in [0,1] \times [0,1]\end{aligned} \tag{18.228}$$

Here, $\delta\mathbf{G}_x = \delta\mathbf{G}_z = \mathbf{0}$ because the change only takes place in the y-direction. Then, the perturbed surface $\mathbf{S}(u,w;\delta\mathbf{G})$ can be obtained as

$$\mathbf{S}(u,w;\delta\mathbf{G}) = \mathbf{S}(u,w) + \delta\mathbf{S}(u,w) \tag{18.229}$$

18.5.3.3 3-D Solid Structures—CAD-Generated Surfaces

In CAD, we sketch an open (or closed) profile and protrude it for a surface or protrude a closed profile for a solid feature (or a surface feature). The basic protrusion capabilities commonly available in CAD include extrusion, blend (or loft), revolve, and sweep. The boundary surface of a solid feature generated in CAD involves moving a curve segment $\mathbf{P}(u)$ (representing the profile of a section sketch) along a certain path $\mathbf{Q}(w)$, where u and w are the parametric coordinates of the resulting parametric surface. The mathematical formulation of the parametric surfaces has been discussed in Chapter 2. In

this subsection, we assume a cylindrical surface generated by extruding a sketch profile along a direction that is perpendicular to the sketch plane.

Mathematically, a cylindrical surface can be written in a parametric form as

$$\mathbf{S}(u,w) = \mathbf{P}(u) + \mathbf{Q}(w) = \mathbf{P}(u) + w\mathbf{r}, \quad (u,w) \in [0,1] \times [0,1] \tag{18.230}$$

in which $\mathbf{P}(u)$ is a curve in the sketch profile, and $\mathbf{Q}(w) = w\mathbf{r}$, $\mathbf{r}$ is the vector of the straight line representing the extrusion direction and depth, as shown in Figure 18.19. In this case, the straight line $\mathbf{r}$ is perpendicular to the sketch plane where the curve $\mathbf{P}(u)$ resides.

A change in the sketch profile or the depth of the extrusion affects the geometric shape of the cylindrical surface. For example, if the curve $\mathbf{P}(u)$ is modeled as a Bézier curve, then the movement of a control point on the sketch plane (e.g., the x–y plane shown in Figure 18.19) alters the geometry of the cylindrical surface by

$$\delta\mathbf{S}(u,w) = \delta\mathbf{P}(u), \quad (u,w) \in [0,1] \times [0,1] \tag{18.231}$$

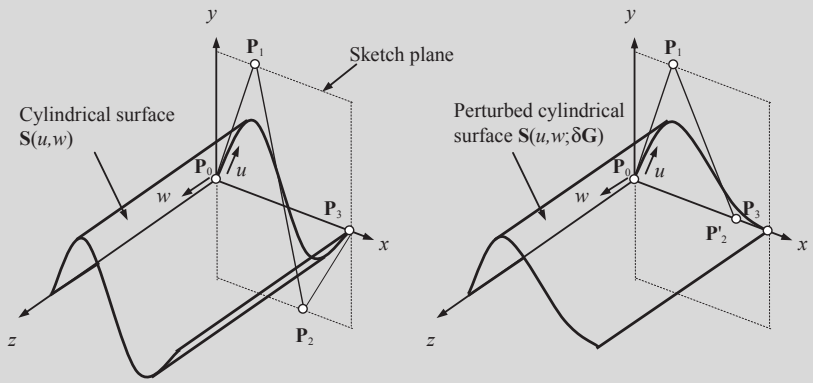
Similarly, if the extrusion depth is changed, the change in the cylindrical surface is

$$\delta\mathbf{S}(u,w) = \delta\mathbf{Q}(w) = w\delta\mathbf{r}, \quad (u,w) \in [0,1] \times [0,1] \tag{18.232}$$

Then, the perturbed surface $\mathbf{S}(u,w;\ \delta\mathbf{G})$ can be obtained as in Eq. 18.229.

EXAMPLE 18.17

Find the parametric equation of the cylindrical surface generated by extruding a cubic Bézier curve on the x–y plane along the positive z-direction for 5 units, as shown below (left). Note that the four points that control the Bézier curve are identical to those of Example 18.16.



If the control point $\mathbf{P}_2$ is moved upward by two units—that is, $\delta P_{2y} = 2$—calculate the change in surface $\delta\mathbf{S}(u,w)$, and calculate the parametric equation of the perturbed cylindrical surface $\mathbf{S}(u,w;\ \delta\mathbf{G})$.

Solution

From Example 18.16, we have the parametric equation of the Bézier curve as

$$\mathbf{P}(u) = \left[3u,\ 15u^3 - 24u^2 + 9u\right] \tag{18.233}$$

EXAMPLE 18.17—cont'd

Using Eq. 18.230, the cylindrical surface can be written as

$$\begin{aligned}\mathbf{S}(u,w) &= \mathbf{P}(u) + w\mathbf{r} = [3u, 15u^3 + 24u^2 + 9u, 0] + w[0,0,5] \\ &= [3u, 15u^3 - 24u^2 + 9u, 5w], \quad (u,w) \in [0,1] \times [0,1]\end{aligned} \tag{18.234}$$

The change in curve due to $\delta P_{2y} = 2$ was obtained in Example 18.16 as

$$\delta\mathbf{P}(u) = [0, -6u^3 + 6u^2]$$

Hence, the change in surface is

$$\delta\mathbf{S}(u,w) = \delta\mathbf{P}(u) = [0, -6u^3 + 6u^2, 0], \quad (u,w) \in [0,1] \times [0,1] \tag{18.235}$$

Then, the perturbed cylindrical surface $\mathbf{S}(u,w;\ \delta\mathbf{G})$ can be obtained using Eq. 18.229 as

$$\begin{aligned}\mathbf{S}(u,w;\delta\mathbf{G}) &= \mathbf{S}(u,w) + \delta\mathbf{S}(u,w) = [3u, 15u^3 - 24u^2 + 9u, 5w] + [0, -6u^3 + 6u^2, 0] \\ &= [3u, 9u^3 - 18u^2 + 9u, 5w], \quad (u,w) \in [0,1] \times [0,1]\end{aligned} \tag{18.236}$$

The cylindrical surface after perturbation is shown above (previous page) to the right.

Note that Eq. 18.230 is not the only way to represent a cylindrical surface mathematically. Another widely employed formulation is based on a polar coordinate system, such as the circular cylindrical surface shown in Figure 18.20. The surface is created by extruding a circular arc along a straight path $\mathbf{d} = w\mathbf{e}_3$, yielding a circular cylindrical surface:

$$\mathbf{S}(u,w) = r \cdot [\cos(u) \cdot \mathbf{e}_1 + \sin(u) \cdot \mathbf{e}_2] + w \cdot \mathbf{e}_3 + \mathbf{o}, \quad (u,w) \in A = [0, \Phi] \times [0,h], \quad \mathbf{o} \in R^3 \tag{18.237}$$

where $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ are orthogonal unit vectors in R^3 and $\mathbf{o}$ is the origin of the circular arc, as shown in Figure 18.20. For this cylindrical surface, design variables can be radius r and height h.

18.5.4 DESIGN VELOCITY FIELD COMPUTATION

When design variables vary, the geometric shape of the structural boundary—and therefore, the location of material points inside the structural domain—must change accordingly. The design velocity field governs the movement of material points both on the boundary and inside the structural domain based on the changes in shape design variables. The design velocity field provides a systematic scheme that maps the location of material points from original design to the updated design. Naturally, design velocity field

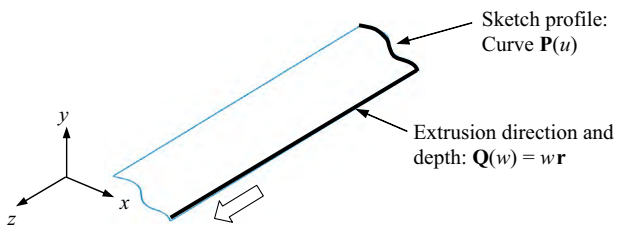


FIGURE 18.19

Extrusion of a sketch profile for a cylindrical surface.

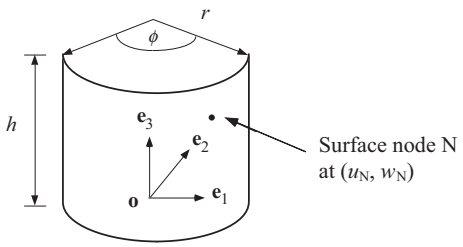


FIGURE 18.20

A parametric cylindrical surface.

supports finite element mesh updates while maintaining mesh topology. Maintaining mesh topology is critical in terms of tracking local measures, such as stress and displacement, during design iterations.

We first define the design velocity mathematically and briefly discuss its theoretical and practical requirements. We present the velocity computation for design boundary, and then domain velocity.

18.5.4.1 Design Velocity Field

Consider a structural domain Ω with its boundary Γ as a continuous medium at the current design $\tau = 0$ shown in Figure 18.21 (solid lines). Suppose only one parameter τ defines the transformation $\mathbf{T}$ that changes the structural domain from Ω to Ω_τ (dotted lines). The transformation mapping $\mathbf{T}$ that represents this process can be defined as

$$\mathbf{T} : \mathbf{x} \rightarrow \mathbf{x}_\tau(\mathbf{x}), \quad \mathbf{x} \in \Omega \tag{18.238}$$

where $\mathbf{x}$ is a material point (Haug et al,. 1986).

We define the design velocity field $\mathbf{V}$ as

$$\mathbf{V}(\mathbf{x}_\tau, \tau) \equiv \frac{d\mathbf{x}_\tau}{d\tau} = \frac{d\mathbf{T}(\mathbf{x}, \tau)}{d\tau} \tag{18.239}$$

where τ plays the role of design time (or the design iteration in practice). In the neighborhood of the current design $\tau = 0$, assume the mapping function $\mathbf{T}$ is smooth. Ignoring higher-order terms, $\mathbf{T}$ can be approximated by

$$\mathbf{T}(\mathbf{x}, \tau) = \mathbf{T}(\mathbf{x}, 0) + \tau \frac{d\mathbf{T}(\mathbf{x}, 0)}{d\tau} + O(\tau^2) \approx \mathbf{x} + \tau \mathbf{V}(\mathbf{x}) \tag{18.240}$$

where $\mathbf{x} \equiv \mathbf{T}(\mathbf{x},0)$ and $\mathbf{V}(\mathbf{x}) \equiv \mathbf{V}(\mathbf{x},0)$. Note that only the linear term is retained in Eq. 18.240.

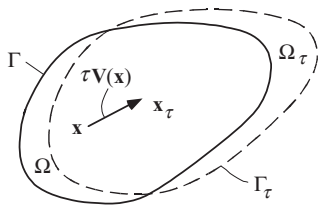


FIGURE 18.21

Changing of the structural domain.

In shape optimization, the design velocity field is calculated first at the design boundary, called the boundary velocity field. The mapping $\mathbf{T}$ is characterized by the parametric equations employed for representing the design boundary. Therefore, the boundary velocity field can be calculated by varying the parametric equations of the design boundary through changes in design variables:

$$\mathbf{V}_i(u) = \delta\mathbf{P}(u;\mathbf{b}) = \frac{\partial \mathbf{P}}{\partial b_i}\delta b_i \tag{18.241a}$$

and

$$\mathbf{V}_i(u,w) = \delta\mathbf{S}(u,w;\mathbf{b}) = \frac{\partial \mathbf{S}}{\partial b_i}\delta b_i \tag{18.241b}$$

where $\mathbf{P}(u)$ and $\mathbf{S}(u,w)$ are the parametric curves and surfaces representing the design boundary of 2-D and 3-D structures, respectively; and $\mathbf{b}$ is the vector of shape design variables. Note that the design perturbation for the ith design variable, δb_i, is usually set to 1 for convenience in practice.

The change of structural boundary also causes the movement of material points in the domain of the structural component, which is characterized by the so-called domain velocity field. Both boundary and domain velocity fields must be calculated. Before discussing the computation methods, there are a few requirements for the design velocity that are worth mentioning (Choi and Chang, 1994).

First, the velocity field must depend linearly on the variation of shape design variables, as required by its definition shown in Eq. 18.240. Linear dependence requires that if control point $\mathbf{P}_1$ moves a distance $\delta b_1 = 1$ in the x-direction, producing domain design velocity $\mathbf{V}^1(\mathbf{x}^1)$ at node 1, as shown in Figure 18.22(a), then node 1 must move $k\mathbf{V}^1(\mathbf{x}^1)$ along the same direction when $\mathbf{P}_1$ moves $k\ \delta b_1 = k$, $k \neq 0$, in the x-direction, as depicted in Figure 18.22(b). This linear dependence must be true for all boundary and interior nodes of the structure.

For a boundary velocity field, as long as the derivatives $\partial\mathbf{P}/\partial b_i$ and $\partial\mathbf{S}/\partial b_i$ shown in Eqs 18.241a and 18.241b, respectively, are constant at a given u and (u,w), the linearity requirement is satisfied.

For a finite element model, design sensitivity coefficients predict structural performance measures of the perturbed design with finite element mesh updated by moving nodal points along the direction of the design velocity field using

$$\mathbf{x}^k(\mathbf{b}+\delta\mathbf{b}) = \mathbf{x}^k(\mathbf{b}) + \delta\mathbf{x}^k(\mathbf{b}) = \mathbf{x}^k(\mathbf{b}) + \sum_{i=1}^{n}\mathbf{V}_i^k\delta b_i \tag{18.242}$$

where $\mathbf{x}^k(\mathbf{b}+\delta\mathbf{b})$ and $\mathbf{x}^k(\mathbf{b})$ are the locations of the kth node of the perturbed and the current designs, respectively; $\delta\mathbf{x}^k(\mathbf{b})$ is the nodal point movement due to design changes; $\mathbf{V}_i^k$ and δb_i are the design

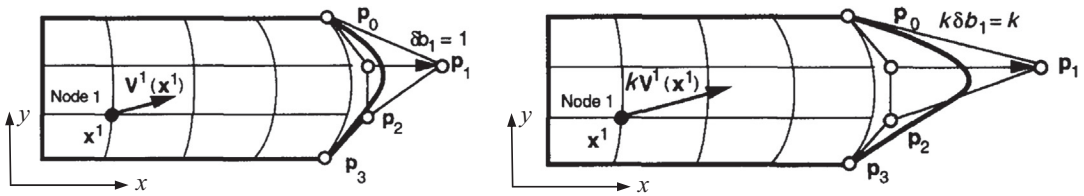


FIGURE 18.22

Illustration of the linearity requirement of the design velocity field. (a) Control point $\mathbf{P}_1$ moves $\delta b_1 = 1$ in the x-direction. (b) Control point $\mathbf{P}_1$ moves $k\delta b_1 = k$, $k \neq 0$ in the x-direction.

velocity of the ith design variable at the kth node and the change of the ith design variable b_i, respectively; and n is the total number of design variables. It is important to note that shape sensitivity coefficients predict performance measures at the new design for the finite element mesh updated using the design velocity field employed for sensitivity analysis.

For practical applications using FEM, a velocity field computation method must retain the topology of the original finite element mesh (i.e., no elements or nodes are added or removed) so that local performance measures, such as displacement or stress, at a specific node are retained consistently throughout the design process. It is also desirable that the finite element mesh that is updated using the design velocity computed does not get distorted in the design process.

18.5.4.2 Boundary Velocity Computation

We discuss boundary velocity computation for 2-D planar and 3-D solid structures.

18.5.4.2.1 2-D Planar Structures

As discussed earlier, the design boundary of a 2-D planar structure is represented using parametric curves. If FEM is employed for structural analysis in shape design, the finite element nodes at the design boundary will have to move to the new geometric boundary at the new design. The movement (i.e., the boundary velocity field) can be calculated by plugging the parametric coordinate u of a boundary node (e.g., u_j of the jth node) on the boundary curve into Eq. 18.241a:

$$\mathbf{V}_i^j = \mathbf{V}_i(u_j) = \delta\mathbf{P}(u_j;\mathbf{b}) = \frac{\partial \mathbf{P}}{\partial \mathrm{b}_i}\delta b_i \tag{18.243}$$

For example, if a cubic Bézier curve is parameterized for the design boundary, the boundary velocity field can be calculated by bringing Eq. 18.216 into Eq. 18.243 as

$$\mathbf{V}_i^j = \mathbf{V}_i(u_j) = \frac{\partial \mathbf{P}(u_j)}{\partial \mathrm{b}_i}\delta \mathrm{b}_i = \mathbf{U}(u_j)\mathbf{N}^{\mathrm{B}}\frac{\partial \mathbf{G}^{\mathrm{B}}}{\partial b_i}\delta b_i \tag{18.244}$$

Here, $\frac{\partial \mathbf{G}^{\mathrm{B}}}{\partial \mathrm{b}_i}$ is $\mathbf{0}$, except for the entry corresponding to a design variable, in which the value is 1. For the design change shown in Figure 18.16(b), in which P_{2y} is varied, we have

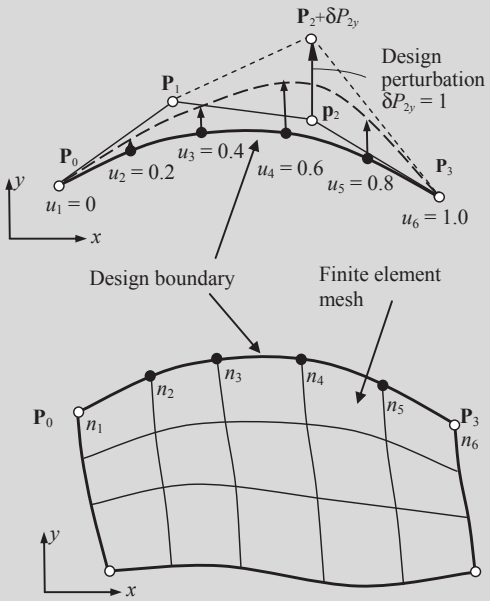
$$\frac{\partial \mathbf{G}^{\mathrm{B}}}{\partial b_i} = \frac{\partial \mathbf{G}^{\mathrm{B}}}{\partial P_{2y}} = \frac{\partial}{\partial P_{2y}}\begin{bmatrix} P_{0x} & P_{0y} \\ P_{1x} & P_{1y} \\ P_{2x} & P_{2y} \\ P_{3x} & P_{3y} \end{bmatrix}_{4\times 2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}_{4\times 2} \tag{18.245}$$

Hence, the boundary velocity for the design variable P_{2y} can be calculated using Eq. 18.244 as

$$\begin{aligned} \mathbf{V}^j = \mathbf{V}(u_j) &= \frac{\partial \mathbf{P}}{\partial P_{2y}}\delta P_{2y} = \mathbf{U}(u_j)\mathbf{N}^{\mathrm{B}}\frac{\partial \mathbf{G}^{\mathrm{B}}}{\partial P_{2y}}\delta P_{2y} \\ &= \begin{bmatrix} u_j^3 & u_j^2 & u_j & 1 \end{bmatrix}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 2 \\ 0 & 0 \end{bmatrix}\delta P_{2y} \end{aligned} \tag{18.246}$$

EXAMPLE 18.18

Given four control points—$\mathbf{P}_0 = [0, 0]$, $\mathbf{P}_1 = [1, 1]$, $\mathbf{P}_2 = [2, 0.5]$, and $\mathbf{P}_3 = [3, 0]$—create a Bézier curve that represents a design boundary of a planar structure meshed with 5×3 finite elements, as shown below. There are six nodes meshed on the design boundary with parametric coordinates $u = 0$, 0.2, 0.4, 0.6, 0.8, and 1.0, respectively. If the vertical movement of the control point $\mathbf{P}_2$ is defined as a design variable—that is, $\delta b = \delta P_{2y} = 1$—then calculate the boundary velocity for nodes 4 and 5 on the design boundary.



Solution

Using Eq. 18.246, we have

$$\mathbf{V}^j = \mathbf{V}(u_j) = \frac{\partial \mathbf{P}}{\partial P_{2y}}\delta P_{2y} = \mathbf{U}(u_j)\mathbf{N}^{\mathrm{B}}\frac{\partial \mathbf{G}^{\mathrm{B}}}{\partial P_{2y}}\delta P_{2y} = \begin{bmatrix} u_j^3 & u_j^2 & u_j & 1 \end{bmatrix}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}\delta P_{2y} \tag{18.247}$$

The parametric coordinate of node 4 is $u_4 = 0.6$. Therefore, the boundary velocity at the node is

$$\mathbf{V}^4 = \mathbf{V}(u_4) = \begin{bmatrix} 0.6^3 & 0.6^2 & 0.6 & 1 \end{bmatrix}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0, 0.432 \end{bmatrix} \tag{18.248}$$

Similarly, $\mathbf{V}^5$ for node 5, where $u_5 = 0.8$, and $\mathbf{V}^5 = [0, 0.384]$.

In practice, implementing Eq. 18.243 for the boundary velocity computation requires knowledge of the parametric coordinates of individual boundary nodes. Usually, only Cartesian coordinates (x,y) of a node are available, such as those found in the finite element input data file. How can we convert x- and

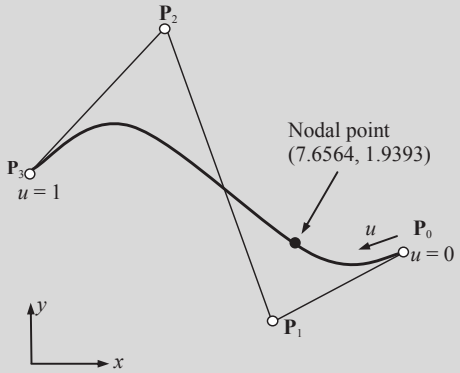
y-coordinates into the parametric coordinate u? One straightforward technique is using MATLAB's root finding functions, such as solve or fzero. We use the following example to illustrate further.

EXAMPLE 18.19

Given four control points—$\mathbf{P}_0 = [10, 2]$, $\mathbf{P}_1 = [6, 1]$, $\mathbf{P}_2 = [4, 5]$, and $\mathbf{P}_3 = [0, 3]$—create a Bézier curve that represents a design boundary of a planar structure. The parametric equation of the Bézier curve is given as

$$\mathbf{P}(u) = \left[-4u^3 + 6u^2 - 12u + 10, -11u^3 + 15u^2 - 3u + 2\right], \quad u \in [0, 1]$$

The Cartesian coordinates of a nodal point on the curve are found in the finite element input data file as $x = 7.6564$ and $y = 1.9393$. Find the parametric coordinate u of the nodal point on the curve.



Solution

We use two MATLAB functions to show the solutions, solve and fzero. We are solving the x-component of the curve equation, as follows:

Find u by solving: $-4u^3 + 6u^2 - 12u + 10 = 7.6564$ (or $-4u^3 + 6u^2 - 12u + 10 - 7.6564 = 0$).

The function `solve`, syntax `solve (eq,x)`, returns the set of all complex solutions of an equation `eq` with respect to x. We pick the real solution as the parametric coordinate for the point. Enter the following in MATLAB:

```
[u]=solve('-4*u^3+6*u^2-12*u+10 = 7.6564')
```

MATLAB returns the following:

```
u =
                    0.21511995841548803773588873485
0.64244002079225598113205563257 5 + 1.52015376806752511784842898177 41*i
0.64244002079225598113205563257 5 - 1.52015376806752511784842898177 41*i
```

The only real solution is `u = 0.21512`.

The function `fzero`, syntax `x=fzero(fun,x0)`, tries to find a point x where `fun(x) = 0`, `x0` is the initial point the user enters. Enter the following in MATLAB:

```
x=fzero('-4*x^3+6*x^2-12*x+10-7.6564',0)
```

MATLAB returns the following:

```
x =
0.2151
```

which is the parametric coordinate we are solving.

18.5.4.2.2 3-D Solid Structures—Freeform Surfaces

Similar to the 2-D applications, boundary velocity field can be calculated by plugging the parametric coordinates at the nodes, such as (u_j, w_j) of the jth node, on the boundary surface into Eq. 18.241b:

$$\mathbf{V}_i^j = \mathbf{V}_i(u_j, w_j) = \delta\mathbf{S}(u_j, w_j; \mathbf{b}) = \frac{\partial \mathbf{S}}{\partial b_i}\delta b_i \tag{18.249}$$

For example, if a bicubic Bézier surface is parameterized for the design boundary, the boundary velocity field can be calculated by bringing Eq. 18.227 into Eq. 18.249:

$$\mathbf{V}_i^j = \mathbf{V}_i(u_j, w_j) = \frac{\partial \mathbf{S}}{\partial b_i}\delta b_i = \mathbf{U}(u_j)\mathbf{N}^{\mathrm{B}}\frac{\partial \mathbf{G}^{\mathrm{B}}}{\partial b_i}\mathbf{N}^{\mathrm{B}}\mathbf{W}(w_j)^{\mathrm{T}}\delta b_i \tag{18.250}$$

in which $\frac{\partial \mathbf{G}^{\mathrm{B}}}{\partial b_i}$ is $\mathbf{0}$, except for the entry corresponding to a design variable, in which the value is 1. For example, if the y-coordinate of the control point $\mathbf{P}_{22}$—that is, P_{22y}—is chosen as a design variable, we have

$$\frac{\partial \mathbf{G}_y^B}{\partial b_i} = \frac{\partial \mathbf{G}_y^B}{\partial P_{22y}} = \frac{\partial}{\partial P_{22y}}\begin{bmatrix} p_{00y} & p_{10y} & p_{20y} & p_{30y} \\ p_{01y} & p_{11y} & p_{21y} & p_{31y} \\ p_{02y} & p_{12y} & p_{22y} & p_{32y} \\ p_{03y} & p_{13y} & p_{23y} & p_{33y} \end{bmatrix}_{4\times 4} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}_{4\times 4} \tag{18.251}$$

where $\frac{\partial \mathbf{G}_x^{\mathrm{B}}}{\partial b_i} = \frac{\partial \mathbf{G}_z^{\mathrm{B}}}{\partial b_i} = 0$ because the change only takes place in the y-direction. Hence, the boundary velocity for the design variable P_{22y} can be calculated using Eq. 18.250:

$$\begin{aligned} \mathbf{V}_i^j = \mathbf{V}_i(u_j, w_j) &= \frac{\partial \mathbf{S}}{\partial P_{22y}}\delta P_{22y} = \mathbf{U}(u_j)\mathbf{N}^{\mathrm{B}}\frac{\partial \mathbf{G}_{\mathrm{y}}^{\mathrm{B}}}{\partial p_{2y}}\mathbf{N}^{\mathrm{B}}\mathbf{W}(w_j)^{\mathrm{T}}\delta P_{2y} \\ &= \begin{bmatrix} u_j^3 & u_j^2 & u_j & 1 \end{bmatrix}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} w_j^3 \\ w_j^2 \\ w_j \\ 1 \end{bmatrix}\delta P_{2y} \end{aligned} \tag{18.252}$$

An important role that freeform surfaces play in shape optimization is that they are suitable for integrating topology optimization (Bendsøe and Sigmund, 2003) with shape optimization because the structural boundary obtained from topology optimization is unsmooth and highly irregular. Freeform surfaces approximate the irregular structural boundary within a prescribed error bound and smooth the boundary for shape optimization. In addition, for follow-up engineering assignments, such as machining simulation using computer-aided manufacturing (CAM) software, the surface parameterization must be compatible with CAD so that the geometry of the component can be imported into CAD for machining simulation and toolpath generation. However, not all freeform surfaces are compatible with CAD systems. In fact, only very few are supported. Among them are Bézier and the B-spline surfaces. Between the two, the B-spline (or NURB) surface is probably the safer choice because it is supported by several major CAD software programs, including SolidWorks. This point is further illustrated in a case study presented in Section 18.7.1.

18.5.4.2.3 3-D Solid Structures—CAD-Generated Surfaces

Similar to the freeform surfaces, the boundary velocity field for a CAD-generated surface can be calculated using Eq. 18.249.

For example, the cylindrical surface shown in Figure 18.19 is parameterized as a design boundary. The boundary velocity field can be calculated by bringing Eq. 18.230 into Eq. 18.249:

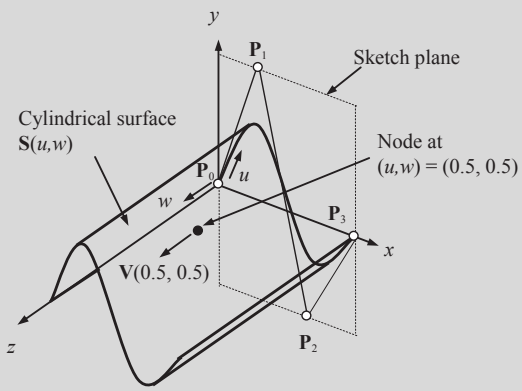
$$\begin{aligned}\mathbf{V}_i(u_j, w_j) &= \frac{\partial \mathbf{S}}{\partial b_i}\delta b_i = \frac{\partial}{\partial b_i}\big(\mathbf{P}(u_j) + \mathbf{Q}(w_j)\big)\delta b_i = \frac{\partial \mathbf{P}(u_j)}{\partial b_i}\delta b_i + \frac{\partial \mathbf{Q}(w_j)}{\partial b_i}\delta b_i \\ &= \mathbf{U}(u_j)\mathbf{N}^{\mathrm{B}}\frac{\partial \mathbf{G}^{\mathrm{B}}}{\partial b_i}\delta b_i + w_j\frac{\partial \mathbf{r}}{\partial b_i}\delta b_i, \quad (u_j, w_j) \in [0,1]\times[0,1]\end{aligned} \tag{18.253}$$

Here, we assume that $\mathbf{P}(u)$ is, for example, a Bézier curve. As with the boundary curve, $\frac{\partial \mathbf{G}^{\mathrm{B}}}{\partial b_i}$ is $\mathbf{0}$, except for the entry corresponding to a design variable, in which the value is 1. If the extrusion depth is the design variable, then $\frac{\partial \mathbf{r}}{\partial b_i} = [0,0,1]$.

EXAMPLE 18.20

Revisiting Example 18.17, the parametric equation of the cylindrical surface shown below is given as follows:

$$\mathbf{S}(u,w) = \mathbf{P}(u) + w\mathbf{r} = \left[3u, 15u^3 - 24u^2 + 9u, 5w\right], \quad (u,w) \in [0,1]\times[0,1] \tag{18.254}$$



If the extrusion depth is defined as the design variable, calculate the change in surface $\delta\mathbf{S}(u,w)$, and calculate the design velocity at a node whose parametric coordinates are (0.5, 0.5).

Solution

From Eq. 18.232, the change in surface $\delta\mathbf{S}(u,w)$ can be calculated as

$$\delta\mathbf{S}(u,w) = \delta\mathbf{Q}(w) = w\delta\mathbf{r} = w[0,0,1]\delta b_i, \quad (u,w) \in [0,1]\times[0,1] \tag{18.255}$$

Then, the velocity at surface node $(u,w) = (0.5, 0.5)$ can be obtained using Eq. 18.253:

$$\mathbf{V}(0.5,0.5) = 0.5\frac{\partial \mathbf{r}}{\partial b_i}\delta b_i = 0.5[0,0,1]\delta b_i = [0,0,0.5] \tag{18.256}$$

Here, δb_i is set to 1, as discussed before.

Similar to the curve boundary, for a surface design boundary, only the Cartesian coordinates (x,y,z) of a node are available, such as those found in the finite element input data file. One may use MATLAB script (or other numerical tools or techniques) to convert (x,y,z) to (u,w). The following illustrates such an example.

EXAMPLE 18.21

The Cartesian coordinates of the surface node $(u,w) = (0.5, 0.5)$ on the surface in Example 18.20 are (1.5, 0.375, 2.5). Convert the Cartesian coordinates of the node back to the parametric coordinates (u,w).

Solution

One possible way to solve for (u,w) from (x,y,z) is using the MATLAB function `fsolve`. We solve (u,w) from the following equations:

$3u = x = 1.5$

$15u^3 - 24u^2 + 9u = y = 0.375$

$5w = 2.5$

This problem is certainly trivial to solve; they are simple, and u and w are decoupled. We pick two equations, `3u-1.5 = 0 and 5w-2.5 = 0`, to illustrate the solution steps.

We first write a file that computes function `F`, the values of the equations at `x`, which is a vector containing u and w:

```
function F = myfun(x)
F = [3*x(1)-1.5;
     5*x(2)-2.5];
```

Save this function file as `myfun.m` somewhere on your MATLAB path. Next, set up the initial point (e.g., `x0 = [0; 0]`) and options and call `fsolve`:

```
x0 = [0;0]; % Make a starting guess at the solution
options = optimoptions('fsolve','Display','iter'); % Option to display output
[x,fval] = fsolve(@myfun,x0,options) % Call solver
```

MATLAB returns the following:

```
Iteration  Func-count  f(x)  Norm of step  First-order optimality  Trust-region radius
0          3          8.5                 12.5                    1
1          6          0     0.707107      0                       1

  Equation solved.
  ...
  x =
        0.5000
        0.5000
  fval =
        0
        0
```

These are the correct parametric coordinates $(u,w) = (0.5, 0.5)$.

18.5.4.2.4 Implementation with CAD Software

The boundary velocity field computation methods discussed above are straightforward to implement because the parametric equations of the boundary curves or surfaces are explicitly expressed in design variables, and numerical data required for evaluating the curves or surfaces are available.

In some situations, however, the design variable is not explicit in the geometric representation for a design surface and the parametric equation, and parametric area A of the perturbed design surface cannot be readily anticipated. Note that in CAD surfaces, (u,w) may not be bound by the parametric area $[0,1] \times [0,1]$. Therefore, an analytical solution for the velocity at a node is not available. In this case, it is easier to let the CAD tool regenerate the part based on the perturbed design variable than try to trace the influence of the design variable based on the surface's geometric representation. In this situation, a viable solution is to use finite difference method to calculate the boundary design velocity. The use of the CAD application protocol interface (API) allows this regeneration to be done automatically. The API also allows one to retrieve the data needed for carrying out velocity field computation.

In the following, we assume a circular cylindrical surface created in Pro/ENGINEER similar to that of Figure 18.20 for discussion. Geometric representations of curves and surfaces in Pro/ENGINEER can be retrieved via the API. Pro/ENGINEER offers API functions that are needed for velocity field computation; for example, pro_get_surface returns a data structure containing the geometric representation of the specified surface, and pro_get_face_params provides the parameter values u and w corresponding to a given point on the surface. These functions are essential for supporting the velocity computation with CAD software, as discussed next.

Suppose the radius r is chosen as a design variable. The parametric equation for the surface is written as $\mathbf{S} = \mathbf{S}(u,w;r)$ to indicate that r is varying. When r is perturbed to $r + \delta r$, the perturbed surface can be represented by

$$\mathbf{S}(u,w;r+\delta r) = (r+\delta r)\cdot[\cos(u)\cdot\mathbf{e}_1 + \sin(u)\cdot\mathbf{e}_2] + (w)\cdot\mathbf{e}_3 + \mathbf{o}, \quad (u,w)\in A = [0,\Phi]\times[0,h] \tag{18.257}$$

On the other hand, suppose h (the height) is chosen as a design variable. Let h be perturbed to $h + \delta h$. Then, the perturbed surface can be represented by

$$\mathbf{S}(u,w;h+\delta h) = r\cdot[\cos(u)\cdot\mathbf{e}_1 + \sin(u)\cdot\mathbf{e}_2] + w\cdot\mathbf{e}_3 + \mathbf{o}, \quad (u,w)\in A' = [0,\Phi]\times[0,h+\delta h] \tag{18.258}$$

where A' denotes the perturbed parametric area. Equivalently, the parametric area may be left unchanged, and the perturbed face represented by

$$\begin{aligned}\mathbf{S}\left(u,\left(1+\frac{\delta h}{h}\right)\cdot w;h\right) &= r[\cos(u)\cdot\mathbf{e}_1 + \sin(u)\cdot\mathbf{e}_2] + \left(1+\frac{\delta h}{h}\right)\cdot w\cdot\mathbf{e}_3 + \mathbf{o}, \quad (u,w)\in A \\ &= [0,\Phi]\times[0,h].\end{aligned} \tag{18.259}$$

Let N be a node on the cylindrical surface with parametric coordinates (u_N, w_N), as shown in Figure 18.20. If r is the design variable, we can calculate the design velocity at N using finite differences:

$$\mathbf{V}^N = \frac{\partial \mathbf{S}}{\partial r} \cong \frac{[\mathbf{S}(u_N,w_N;r+\delta r) - \mathbf{S}(u_N,w_N;r)]}{\delta r} \tag{18.260}$$

Note that, in this case, $\mathbf{V}^N = \cos(u_N)\,\mathbf{e}_1 + \sin(u_N)\,\mathbf{e}_2$. If h is the design variable, the design velocity at N will be

$$\mathbf{V}^N = \frac{\partial \mathbf{S}}{\partial h} \cong \frac{\left[\mathbf{S}\left(u_N, \left(1 + \frac{\delta h}{h}\right) \cdot w_N; r\right) - \mathbf{S}(u_N, w_N; r)\right]}{\delta h} \tag{18.261}$$

In this case, $\mathbf{V}^N = (w_N/h)\,\mathbf{e}_3$. This is the basic approach to computing the design velocity for all nodes on a design surface.

Note that the design perturbation for velocity field computation in CAD must be determined based on the size of the solid features and components. No topological change is allowed in such a perturbation. In this cylindrical surface example, the boundary design velocity field for the two types of perturbations is linearly dependent upon the variation of the design variable. This is so because the parametric equation $\mathbf{S}$ was linear with respect to r and to h. In many situations, the linearity may not be possible (Hardee et al., 1999).

18.5.4.3 Domain Velocity Computation

Perturbation of shape design variables results in the movement of the design boundary, as discussed in the previous section. Interior material points, or finite element nodes, in the structural domain must also move following certain rules. Among numerous approaches for domain velocity field computations, we briefly discuss two representative methods: the boundary displacement method and the isoparametric mapping method (Choi and Chang, 1994).

18.5.4.3.1 Boundary Displacement Method

In the boundary displacement method, movements of nodal points at the design boundary are treated as the prescribed displacements. The domain velocity field that corresponds to the design perturbation can be obtained by solving an auxiliary elasticity problem with prescribed displacements specified at design boundary nodes as well as prescribed rollers at the nondesign boundary, as depicted in Figure 18.23. Note that rollers are added to constrain the nodal point movements at the nondesign boundary, as shown in Figure 18.23(b), in accordance with the design intent.

To form the auxiliary problem for the domain velocity computation, both the velocity field of the design boundary and the displacement constraints (e.g., rollers) that define nodal point movements on the nondesign boundary must be imposed to the finite element model. The discretized equilibrium equation of the auxiliary finite element model can be written as

$$\mathbf{KV} = \mathbf{f} \tag{18.262}$$

where $\mathbf{K}$ is the reduced stiffness matrix of the auxiliary structure (e.g., the structure shown in Figure 18.23(b)), which is different from that of the original structure (e.g., the one shown in Figure 18.23(a)). $\mathbf{V}$ is the design velocity vector and $\mathbf{f}$ is the unknown vector of boundary forces, which produce the prescribed boundary velocity at both the design and nondesign boundary. In a partitioned form, Eq. 18.262 can be written as

$$\begin{bmatrix} \mathbf{K}_{bb} & \mathbf{K}_{db} \\ \mathbf{K}_{db} & \mathbf{K}_{dd} \end{bmatrix} \begin{bmatrix} \mathbf{V}_b \\ \mathbf{V}_d \end{bmatrix} = \begin{bmatrix} \mathbf{f}_b \\ 0 \end{bmatrix} \tag{18.263}$$

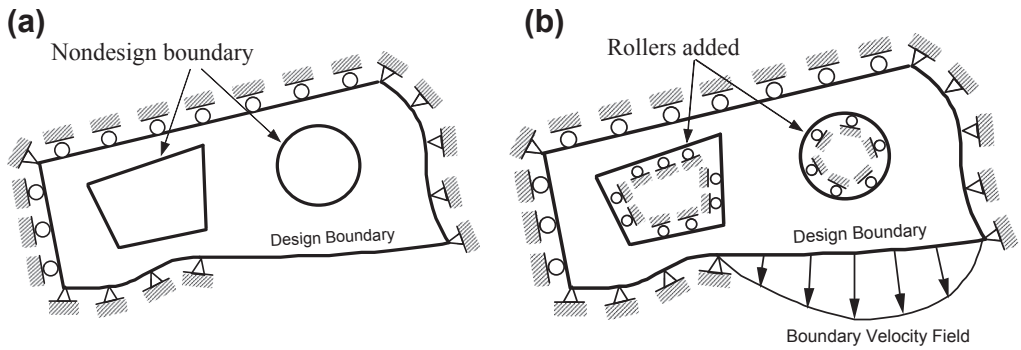


FIGURE 18.23

Illustration of the boundary displacement method for domain velocity field computation. (a) The original structure with boundary conditions. (b) The boundary velocity field added to the design boundary and additional rollers added to the nondesign boundary.

where $\mathbf{V}_b$ is the prescribed velocity of nodes on the boundary, $\mathbf{V}_d$ is the nodal velocity vector in the interior domain (domain velocity), and $\mathbf{f}_b$ is the unknown boundary force that acts on the structural boundary. Equation 18.263 can be rearranged as

$$\mathbf{K}_{dd}\mathbf{V}_d = -\mathbf{K}_{db}\mathbf{V}_b \tag{18.264}$$

This defines a linear relationship between the boundary and domain velocity fields.

The boundary displacement method is independent of the way in which the boundary velocity field is computed. The same finite element code used for analysis can be used to compute the displacement field of the auxiliary model; yielding a domain velocity field that naturally satisfies the linearity requirement (because the finite element matrix equations are a system of linear equations). An important characteristic of the elasticity problem is that the solution trajectory tends to maintain orthogonality. As a result, the updated mesh obtained using the velocity field, as stated in Eq. 18.242, tends to be more regular (see Figure 18.26 as an example).

One drawback of the boundary displacement method is that finite element matrix equations must be formulated and solved to generate the velocity field for each shape design variable. For this, the reduced stiffness matrix of the auxiliary finite element model must be formed and decomposed for each design variable. Once the design velocity field is calculated, it can be used throughout the design iterations.

18.5.4.3.2 Isoparametric Mapping Method

The isoparametric mapping method is far more efficient than the boundary displacement method because the former needs only a few matrix multiplications. However, the essence of the isoparametric method is the availability of a mapping function **N** that maps nodal points from parametric coordinates (u, w) of the geometric model to Cartesian coordinates (x, y, z), as illustrated in Figure 18.24.

In Figure 18.24, the structural domain is meshed into 5×3 finite elements. If the nodes are evenly distributed in the (u, w) space, the parametric coordinates of individual nodes are readily available. For example, the parametric coordinates of node 8 are $(u_8, w_8) = (0.2, 0.667)$. To simplify the mathematical

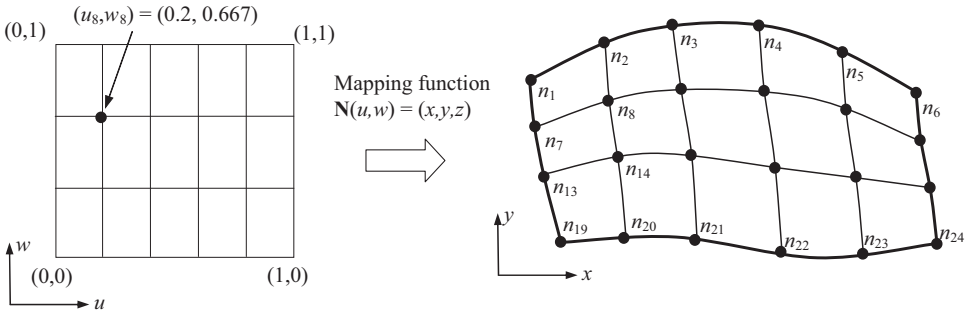


FIGURE 18.24

Concept of isoparametric mapping.

expressions in our discussion, we assume a 2-D planar structure for discussing the parametric mapping method.

Finding an appropriate mapping function **N** for accurate velocity field computation is difficult when the geometric shape of the structure is complicated. However, for a simple geometric model or a smaller subdomain of the structure, for which an accurate mapping function **N** can be found, the isoparametric mapping method is attractive. For such a model or subdomain that can be modeled by a single geometric entity, velocity fields can be computed using the isoparametric mapping method.

Geometric modeling software, such as MSC/PATRAN and HyperMesh, employs a standard patch to represent geometric surfaces. For example, in PATRAN, a surface patch is modeled in Coons patch (Mortenson, 2006). A Coons patch, as shown in Chapter 2, Figure 2.17(b), is a bicubic parametric surface in terms of parametric coordinates u and w, where u and $w \in [0,1]$. Obviously, one single patch is not able to represent a complicate geometric entity. Therefore, a 2-D structural domain is often decomposed into several smaller patches in the modeling process.

The edges of a Coons patch are cubic curves in u and w, respectively. The shape of the cubic curve can be controlled by adjusting tangent vectors in both direction and magnitude. For example, changing the direction of the tangent vector $\mathbf{P}_{11,u}$ will alter the geometry of edge 3 and, therefore, the geometry of the patch, as illustrated in Chapter 7, Figure 7.34(b). A mesh of quad-elements can be generated by specifying the number of elements along the u- and w-parametric directions, such as the mesh shown in Figure 18.24, in which the number of elements along the u- and w-parametric directions are 5 and 3, respectively.

The data representation of a Coons patch is in a $4\times4\times2$ -matrix, as discussed in Chapters 2 and 7 (see Eq. 2.70 and Figure 7.34(c)), defined as

$$\mathbf{G}^{\mathrm{V}} = \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{00,w} & \mathbf{P}_{01,w} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{10,w} & \mathbf{P}_{11,w} \\ \mathbf{P}_{00,u} & \mathbf{P}_{01,u} & \mathbf{P}_{00,uw} & \mathbf{P}_{01,uw} \\ \mathbf{P}_{10,u} & \mathbf{P}_{11,u} & \mathbf{P}_{10,uw} & \mathbf{P}_{11,uw} \end{bmatrix}_{4\times4\times2} \tag{18.265}$$

Recall that the mathematical expression (algebraic format) for a bicubic parametric surface was given in Eq. 18.226. Similar to a Bézier surface, a Coons patch can also be written in a form like that of Eq. 18.227:

$$\begin{aligned}\mathbf{S}(u,w) &= \mathbf{U}\mathbf{N}^{\mathrm{V}}\mathbf{G}^{\mathrm{V}}\mathbf{N}^{\mathrm{V}^{\mathrm{T}}}\mathbf{W}^{\mathrm{T}} \\ &= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} & \mathbf{P}_{00,w} & \mathbf{P}_{01,w} \\ \mathbf{P}_{10} & \mathbf{P}_{11} & \mathbf{P}_{10,w} & \mathbf{P}_{11,w} \\ \mathbf{P}_{00,u} & \mathbf{P}_{01,u} & \mathbf{P}_{00,uw} & \mathbf{P}_{01,uw} \\ \mathbf{P}_{10,u} & \mathbf{P}_{11,u} & \mathbf{P}_{10,uw} & \mathbf{P}_{11,uw} \end{bmatrix}_{4\times4\times2} \\ &\quad \times \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix} \end{aligned} \tag{18.266}$$

Here, $(u,w) \in [0,1] \times [0,1]$, and as discussed in Chapter 2 (see Eq. 2.69)

$$\mathbf{N}^{\mathrm{V}} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{18.267}$$

The boundary edges of a Coons patch are Hermit cubic curves, defined as

$$\mathbf{P}(u) = \mathbf{U}\mathbf{N}^{\mathrm{V}}\mathbf{H} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_{0,u} \\ \mathbf{P}_{1,u} \end{bmatrix} \tag{18.268}$$

Here, $\mathbf{H} = [\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_{0,u}, \mathbf{P}_{1,u}]^{\mathrm{T}}$, in which $\mathbf{P}_0$ and $\mathbf{P}_1$ are the end points and $\mathbf{P}_{0,u}$ and $\mathbf{P}_{1,u}$ are tangent vectors at the end points. Entries in the matrix $\mathbf{H}$ can be used for shape design variables, depending on which boundary edge the curve represents. Any change in $\mathbf{H}$ alters the shape of the Coons patch:

$$\delta\mathbf{S}(u,w) = \mathbf{U}\mathbf{N}^{\mathrm{V}}\delta\mathbf{G}^{\mathrm{V}}\mathbf{N}^{\mathrm{V}^{\mathrm{T}}}\mathbf{W}^{\mathrm{T}} \tag{18.269}$$

where

$$\delta\mathbf{G}^{\mathrm{V}} = \begin{bmatrix} \delta\mathbf{P}_{00} & \delta\mathbf{P}_{01} & \delta\mathbf{P}_{00,w} & \delta\mathbf{P}_{01,w} \\ \delta\mathbf{P}_{10} & \delta\mathbf{P}_{11} & \delta\mathbf{P}_{10,w} & \delta\mathbf{P}_{11,w} \\ \delta\mathbf{P}_{00,u} & \delta\mathbf{P}_{01,u} & 0 & 0 \\ \delta\mathbf{P}_{10,u} & \delta\mathbf{P}_{11,u} & 0 & 0 \end{bmatrix}_{4\times4\times2} \tag{18.270}$$

For example, if the y-component of the tangent vector along the u-direction at $(u,w) = (1,1)$—that is, $P_{11y,u}$—is varied by $\delta P_{11y,u} = 1$, then $\delta\mathbf{G}^{\mathrm{V}}$ is

$$\delta\mathbf{G}_y^{\mathrm{V}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{4\times 4} \tag{18.271}$$

In this case, $\delta\mathbf{G}_x^{\mathrm{V}}(u,w) = 0$ because no change occurs in the x-direction. The domain velocity can be calculated by plugging the (u,w) of the node into Eq. 18.269. Note that in Eq. 18.270, we set the variation of the twister vectors to zero for planar structures.

EXAMPLE 18.22

Assume that $\delta P_{11y,u} = 1$. Calculate the velocity for node 8 (n_8) shown in Figure 18.24, where $(u_8, w_8) = (0.2, 0.667)$.

Solution

Using Eq. 18.269 with $\delta\mathbf{G}_y^{\mathrm{V}}$ shown in Eq. 18.271, we have

$$\begin{aligned} V_y^8 &= \delta \mathrm{S}_y(u_8, w_8) = \delta \mathrm{S}_y(0.2, 0.667) \\ &= \begin{bmatrix} 0.2^3 & 0.2^2 & 0.2 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}_{4\times 4} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.667^3 \\ 0.667^2 \\ 0.667 \\ 1 \end{bmatrix} = -0.0237 \end{aligned}$$

Hence, the velocity is $\mathbf{V}^8 = [0, -0.0237]$.

If the boundary curve is not parameterized using a Hermit cubic curve, the change in the curve must be transformed to a Hermit cubic form, as discussed in Chapter 2, and then the perturbed matrix $\delta\mathbf{G}^{\mathrm{V}}$ is created accordingly. For example, a cubic curve defined in Eq. 18.216 can be written as Bézier curve or a Hermit cubic curve, as follows:

$$\mathbf{P}(u) = \mathbf{U}(u)\mathbf{A} = \mathbf{U}\mathbf{N}^{\mathrm{B}}\mathbf{G}^{\mathrm{B}} = \mathbf{U}\mathbf{N}^{\mathrm{V}}\mathbf{H} \tag{18.272}$$

Here, $\mathbf{G}^{\mathrm{B}}$ contains the four control points of the cubic Bézier curve, and $\mathbf{H}$ was defined in Eq. 18.268, consisting of the end points and tangent vectors of a Hermit cubic curve. Therefore, control point vector of a Bézier curve can be transformed into an $\mathbf{H}$ vector of a Hermit cubic curve as

$$\mathbf{H} = \left(\mathbf{N}^{\mathrm{V}}\right)^{-1}\mathbf{N}^{\mathrm{B}}\mathbf{G}^{\mathrm{B}} \tag{18.273}$$

Then, the changes in a Bézier curve can be transformed into a change in the Hermit cubic curve:

$$\delta\mathbf{H} = \left(\mathbf{N}^{\mathrm{V}}\right)^{-1}\mathbf{N}^{\mathrm{B}}\delta\mathbf{G}^{\mathrm{B}} \tag{18.274}$$

EXAMPLE 18.23

Assume that the design boundary of the patch that contains nodes n_1 to n_6 shown in Figure 18.24 is parameterized by a Bézier curve identical to that of Example 18.18. As stated in Example 18.18, the design change is $\delta b = \delta P_{2y} = 1$; that is, the design variable is the y-direction movement of control point $\mathbf{P}_2$. Note that the design boundary is edge 3 of a Coons patch shown in Chapter 7, Figure 7.34(b). Calculate the velocity for node 8 (see Figure 18.24) inside the patch.

Solution

From Example 18.18, the $\delta\mathbf{G}_y^{\mathrm{B}}$ of the Bézier curve is

$$\delta\mathbf{G}_y^{\mathrm{B}} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \delta P_{2y} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{18.275}$$

From Eq. 18.274, we have

$$\delta\mathbf{H}_y = \left(\mathbf{N}^{\mathrm{V}}\right)^{-1}\mathbf{N}^{\mathrm{B}}\delta\mathbf{G}_y^{\mathrm{B}} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -3 \\ 0 \end{bmatrix} \tag{18.276}$$

Using Eq. 18.269 with the second column of $\delta\mathbf{G}_y^{\mathrm{V}}$ in Eq. 18.271 (with edge 3 as the design boundary in this case) replaced by the vector $\delta\mathbf{H}_y$ in Eq. 18.276, we have

$$\mathrm{V}_y^8 = \delta\mathrm{S}_y(u_8, w_8) = \delta\mathrm{S}_y(0.2, 0.667)$$

$$= \begin{bmatrix} 0.2^3 & 0.2^2 & 0.2 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}_{4\times4} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.667^3 \\ 0.667^2 \\ 0.667 \\ 1 \end{bmatrix} = -0.2846$$

Hence, the domain velocity at node 8 is obtained as $\mathbf{V}^8 = [0, -0.2846]$.

As mentioned before, the isoparametric mapping method is much more efficient than the boundary displacement method; however, the velocity field obtained using the boundary displacement method tends to keep the finite element mesh more regular because trace of the solution to an elastic problem, governed by partial differential equations, is more orthogonal. Figure 18.25(a) shows a 2-D fillet example, in which the finite element mesh updated using the velocity obtained from the boundary displacement method (Figure 18.25(b)) is of better quality than those updated using the isoparametric mapping method. Figure 18.25(c) shows that, using isoparametric mapping, the finite element mesh could be severely distorted after a large design change (e.g., element A).

18.5.5 SHAPE SENSITIVITY ANALYSIS USING FINITE DIFFERENCE OR SEMI-ANALYTICAL METHOD

As with the sensitivity analysis for sizing or material design variables, shape DSA can be carried out using the overall finite difference or semi-analytical method, in which a design variable is perturbed and a finite element model is regenerated with the design change. These methods are general and widely employed in support of gradient-based optimization, despite the drawbacks pointed out previously.

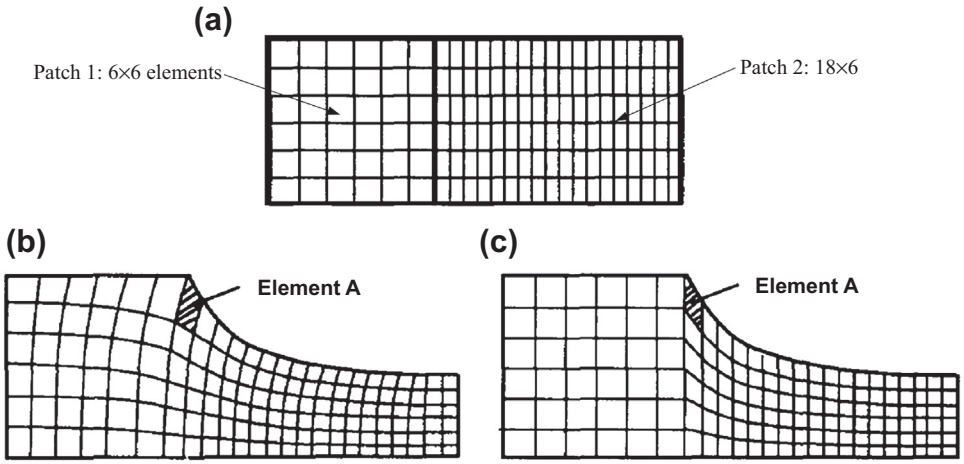


FIGURE 18.25

The two-dimensional fillet example modeled in two patches. (a) Original mesh. (b) Remesh using the boundary displacement method. (c) Remesh the using mapping method (Choi and Chang, 1994).

For shape sensitivity analysis using the finite difference method or semi-analytical method, it is desirable to update the finite element model (using Eq. 18.242) at the perturbed design using a velocity field obtained from the methods discussed above. In the overall finite difference method, an FEA is carried out for the FE model at the perturbed design, then the sensitivity coefficient is approximated using Eq. 18.35. In the semi-analytical method, the stiffness matrix and load vector of the perturbed design must be generated, and Eq. 18.45 is followed to calculate the sensitivity coefficients.

In many situations, the design velocity field may not be readily available. Overall, the finite difference method is widely implemented for shape sensitivity analysis using CAD software equipped with a mesh generator and FEA solver, such as SolidWorks Simulation. A new dimension value with a design perturbation can be entered in CAD and the solid model can be rebuilt and remeshed. Thereafter, an FEA model for the perturbed design can be created (usually meshed by an automatic mesh generator), and an FEA can be carried out. The results obtained from FEA of the current and perturbed designs can then be used for sensitivity calculations using Eq. 18.35.

Although the approach is easy to implement, it has two potential pitfalls in addition to the problems of step size determination and computation efficiency, as discussed in Section 18.3. For shape design, the added pitfalls are altering topology of finite element mesh or geometric features at the perturbed design, as illustrated in Figure 18.26.

The pitfalls affect local performance measures, such as the stress measure defined at node A shown in Figure 18.26(a). We assume a design change pulls the holes further apart. As a result, mesh is regenerated due to a design perturbation, as shown in Figure 18.26(b). Due to the change in mesh topology, there is no clear trace in locating node A in the perturbed design: Is it at node A1, A2, or A3 that the stress performance is to be measured for sensitivity calculation using finite difference method? Another such example is the disappearing of nodes where performance measures are defined. As shown in Figure 18.26(c), a performance measure, such as displacement, is defined at node B. Due

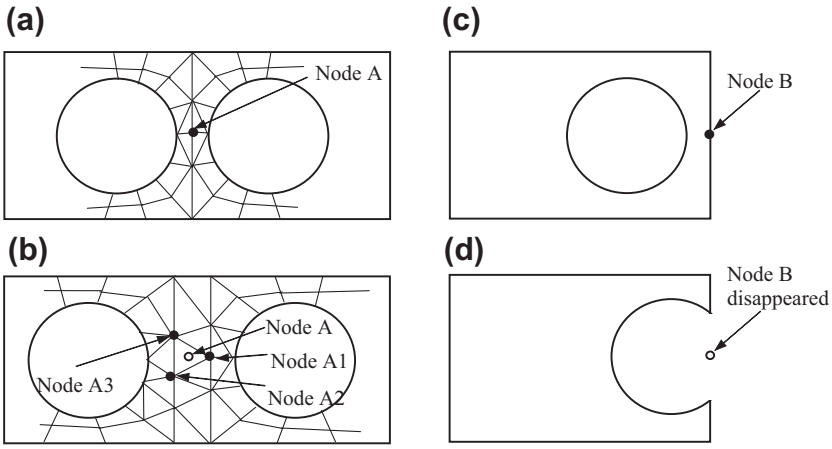


FIGURE 18.26

Pitfalls in shape sensitivity analysis using finite difference or semi-analytical methods with the mesh generator in CAD software: (a) mesh of current design, (b) mesh topology altered after design change, (c) node B at the right edge of the current design, and (d) topology of geometric features altered eliminating node B.

to a design perturbation, the hole is moved to the right, such that it intersects the right edge of the rectangular structural boundary. As a result, part of the right edge disappears, as well as node B. Such a topology change in geometric features causes problems in calculating shape sensitivity coefficients using the finite difference method with CAD software.

18.5.6 MATERIAL DERIVATIVES

Besides finite difference and semi-analytical methods, the continuum approach offers options for shape sensitivity analysis with a rigorous mathematical basis and better computational efficiency. Moreover, no step size is required for design perturbation, such as in the finite difference and semi-analytical methods.

The key theory in the continuum approach for shape sensitivity analysis is the material derivative. In general, the material derivative describes the time rate of change of a physical quantity, such as heat or momentum, for a material element subjected to a space- and time-dependent velocity field. For structural shape design, time is replaced by design change, and the time rate of change of a physical quantity becomes the rate of a performance measure change with respect to design (i.e., gradient or sensitivity coefficient).

Recall the energy bilinear form and load linear form discussed in Section 18.4.1, which governs the behavior of the structure under static load:

$$a_\Omega(\mathbf{z}, \bar{\mathbf{z}}) = \ell_\Omega(\bar{\mathbf{z}}), \quad \forall \bar{\mathbf{z}} \in S \tag{18.277}$$

Note that the subscript Ω is added to the energy and load forms in Eq. 18.277 to emphasize their dependence on the structural domain Ω, which varies. In Eq. 18.277, $\mathbf{z}$ is the solution, or displacement, of the governing equation of the structure at the current design, in which the structural domain is Ω, as illustrated in Figure 18.27.

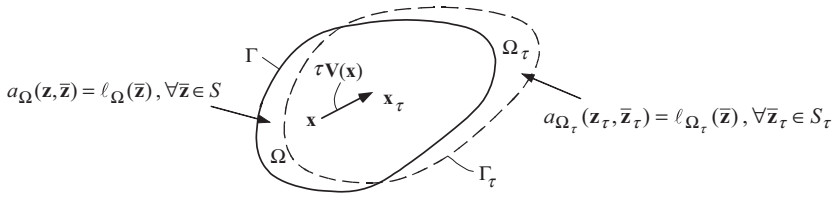


FIGURE 18.27

Illustration of material derivative concept.

Let $\mathbf{z}_\tau(\mathbf{x}_\tau)$ be the solution of the same governing equation but at a perturbed domain Ω_τ due to a design change:

$$a_{\Omega_\tau}(\mathbf{z}_\tau, \bar{\mathbf{z}}_\tau) = \ell_{\Omega_\tau}(\bar{\mathbf{z}}), \quad \forall \bar{\mathbf{z}}_\tau \in S_\tau \tag{18.278}$$

Here, the energy bilinear form and load linear form are formulated at the new structural domain Ω_τ, and S_τ is the space of kinematically admissible virtual displacement at the new design. Note that, in general, $S = S_\tau$ because the essential boundary conditions do not change with design and the smoothness of the solution function is not affected by the shape change. Again, τ is a scalar parameter that defines a shape change. In a material derivative for a physical problem, τ plays the role of time.

It is important to note that the displacement $\mathbf{z}_\tau(\mathbf{x}_\tau)$ at the new design is a new function $\mathbf{z}_\tau$ and measured at the new location $\mathbf{x}_\tau$, which is determined by the design velocity field $\mathbf{V}$, as discussed in Section 18.5.4. As discussed before, for a linear design velocity field, $\mathbf{x}_\tau$ is determined by

$$\mathbf{x}_\tau = \mathbf{x} + \tau\mathbf{V} \tag{18.279}$$

Expanding $\mathbf{z}_\tau(\mathbf{x}_\tau)$ using Taylor's series, we have

$$\mathbf{z}_\tau(\mathbf{x}_\tau) = \mathbf{z}_\tau(\mathbf{x} + \tau\mathbf{V}) \approx \mathbf{z}_\tau(\mathbf{x}) + \nabla\left[\mathbf{z}_\tau^{\mathrm{T}}(\mathbf{x})\right](\tau\mathbf{V}) + \ldots \tag{18.280}$$

Here, the gradient operator is employed—that is, $\nabla = [\partial/\partial \mathrm{x}_1, \partial/\partial \mathrm{x}_2, \partial/\partial \mathrm{x}_3]^{\mathrm{T}}$, which is a 3×1 vector for three-dimensional structures. The material derivative of a function, such as the displacement, is defined as

$$\dot{\mathbf{z}}(\mathbf{x}) \equiv \left.\frac{\mathrm{d}\mathbf{z}_\tau(\mathbf{x}_\tau)}{\mathrm{d}\tau}\right|_{\tau=0} = \left.\frac{\mathrm{d}\mathbf{z}_\tau(\mathbf{x}+\tau\mathbf{V})}{\mathrm{d}\tau}\right|_{\tau=0} \tag{18.281}$$

in which the dot on top of the function $\mathbf{z}$ denotes the material derivative on $\mathbf{z}$. Equation 18.281 can be rewritten in a limiting form as

$$\dot{\mathbf{z}}(\mathbf{x}) \equiv \left.\frac{\mathrm{d}\mathbf{z}_\tau(\mathbf{x}+\tau\mathbf{V})}{\mathrm{d}\tau}\right|_{\tau=0} = \lim_{\tau\to 0}\frac{\mathbf{z}_\tau(\mathbf{x}+\tau\mathbf{V}) - \mathbf{z}(\mathbf{x})}{\tau} = \lim_{\tau\to 0}\frac{\mathbf{z}_\tau(\mathbf{x}+\tau\mathbf{V}) - \mathbf{z}_\tau(\mathbf{x})}{\tau} + \lim_{\tau\to 0}\frac{\mathbf{z}_\tau(\mathbf{x}) - \mathbf{z}(\mathbf{x})}{\tau} \tag{18.282}$$

The first term on the right-hand side of Eq. 18.282 defines the change of the same function $\mathbf{z}_\tau$ evaluated at different locations ($\mathbf{x}_\tau$ and $\mathbf{x}$, respectively). This can be further derived using the expansion shown in Eq. 18.280 as

$$\lim_{\tau\to 0}\frac{\mathbf{z}_\tau(\mathbf{x}+\tau\mathbf{V}) - \mathbf{z}_\tau(\mathbf{x})}{\tau} = \lim_{\tau\to 0}\frac{\nabla\left[\mathbf{z}_\tau^{\mathrm{T}}(\mathbf{x})\right](\tau\mathbf{V}) + \ldots}{\tau} = \nabla\left(\mathbf{z}^{\mathrm{T}}\right)\mathbf{V} = \nabla\mathbf{z}^{\mathrm{T}}\mathbf{V} \tag{18.283}$$

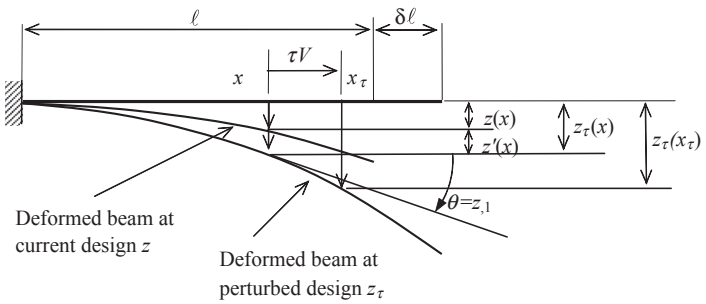


FIGURE 18.28

Deformed cantilever beam at current and perturbed designs due to a length change.

The second term on the right-hand side of Eq. 18.282 defines the change due to different functions ($\mathbf{z}_\tau$ and $\mathbf{z}$) evaluated at the same location $\mathbf{x}$, which is nothing but the variation of $\mathbf{z}$ from its definition discussed in Section 18.4.2:

$$\lim_{\tau \to 0} \frac{\mathbf{z}_\tau(\mathbf{x}) - \mathbf{z}(\mathbf{x})}{\tau} = \mathbf{z}' \tag{18.284}$$

Hence, Eq. 18.282 becomes

$$\dot{\mathbf{z}}(\mathbf{x}) = \mathbf{z}' + \boldsymbol{\nabla}\mathbf{z}^{\mathrm{T}}\mathbf{V} \tag{18.285}$$

The physical meaning of Eq. 18.285 can be illustrated using a cantilever beam example shown in Figure 18.28, in which $\mathbf{z} = z_3 = z$ (subscript 3 is omitted for simplicity), $\mathbf{x} = x_1 = x$ (subscript 1 is omitted), and $\mathbf{V} = V_1 = V$ (subscript 1 is omitted). Hence, Eq. 18.285 is reduced to

$$\dot{z}(x) = z' + z_{,1}V \tag{18.286}$$

in which V is the design velocity that determines the material point movement due to the beam length change. As shown in Figure 18.28, the difference between $z(x)$ and $z_\tau(x_\tau)$ consists of two parts, $z'(x) \approx z_\tau(x) - z(x)$ and $\theta \cdot (x_\tau - x) = z_{,1}\,\tau V$.

Recall that the FEA solution for the cantilever beam with an evenly distributed load shown in Figure 18.13 is obtained in Eq. 18.200 as

$$z_3(x) = \frac{q\ell}{24EI}\left(5x^2\ell - 2x^3\right) \tag{18.287}$$

In the next example, we use this solution to further illustrate the concept of material derivatives for the cantilever beam example shown in Figure 18.13.

EXAMPLE 18.24

Calculate $\dot{z}$, z', and $z_{,1}V$ at $x = \ell$ for the cantilever beam shown in Figure 18.13, using the finite element solution of Eq. 18.200.

Solution

We first evaluate z' from the definition stated in Eq. 18.284. The displacement at the perturbed design due to the change $\delta\ell$ in the length of beam can be found as

$$z_\tau(x) = \frac{q(\ell + \tau\delta\ell)}{24EI}\left(5x^2(\ell + \tau\delta\ell) - 2x^3\right) = \frac{q}{24EI}\left(5x^2(\ell + \tau\delta\ell)^2 - 2x^3\left(\ell + \tau\delta\ell\right)\right) \tag{18.288}$$

EXAMPLE 18.24—cont'd

Hence from Eq. 18.284, z' can be calculated as

$$z' = \lim_{\tau \to 0} \frac{z_\tau(x) - z(x)}{\tau} = \frac{\frac{q}{24EI}\left(5x^2(\ell + \tau\delta\ell)^2 - 2x^3(\ell + \tau\delta\ell)\right) - \frac{q}{24EI}\left(5x^2\ell^2 - 2x^3\ell\right)}{\tau}$$
$$= \lim_{\tau \to 0} \frac{q}{24EI} \frac{2x^2(5\ell - x)\tau\delta\ell + 5x^2(\tau\delta\ell)^2}{\tau} = \frac{qx^2}{12EI}(5\ell - x)\delta\ell \tag{18.289}$$

For simplicity in shape sensitivity analysis, we set $\delta\ell = 1$. Note that Eq. 18.289 is nothing but $\frac{\partial z_3(x)}{\partial \ell}$ shown in Eq. 18.208. Evaluating Eq. 18.289 at $x = \ell$, we have

$$z' = \frac{q\ell^3}{3EI} \tag{18.290}$$

which is identical to Eq. 18.209. Now, we take the derivative of Eq. 18.200 with respect to x:

$$z_{,1}(x) = \frac{q\ell}{12EI}\left(5x\ell - 3x^2\right) \tag{18.291}$$

At $x = \ell$, $V = \delta\ell$. Hence,

$$z_{,1}V = \frac{q\ell^3}{6EI}\delta\ell = \frac{q\ell^3}{6EI} \tag{18.292}$$

Adding Eq. 18.290 and 18.292, we have

$$\dot{z}(x) = z' + z_{,1}V_1 = \frac{q\ell^3}{3EI} + \frac{q\ell^3}{6EI} = \frac{q\ell^3}{2EI} \tag{18.293}$$

which is $\frac{\partial z_3^\ell}{\partial \ell} = \frac{q\ell^3}{2EI}$, as obtained in Eq. 18.202 (and Eq. 18.207).

As demonstrated in Example 18.24, for the cantilever beam example, the sensitivity coefficient obtained using the discrete approach discussed at the beginning of this section is the same as the result of the material derivative, in which the measurement point is assumed to be moving with the design. On the other hand, the result obtained in Eq. 18.209 is nothing but z', in which the measurement point is assumed to be stationary. As discussed, the movement of the material point or the measurement point is determined by the design velocity field. At the boundary, the velocity field is $\delta\ell$. How do the interior points move due to the design change $\delta\ell$? How does the domain velocity field affect the shape sensitivity coefficient?

Now, we discuss further on the design velocity field. At the tip, the boundary velocity is defined as $\delta\ell$. The interior or domain velocity can be defined in different ways, depending on the design intent. For example, let us consider two design velocity fields V^{A} and V^{B}, shown in Figures 18.29(a) and (b), respectively. V^{A} is a linear function, and V^{B} is a quadratic function. Both assume a length change at the right end by $\delta\ell$. Mathematically, they are defined as

$$V^{\mathrm{A}} = \frac{x}{\ell}\delta\ell \tag{18.294}$$

$$V^{\mathrm{B}} = \left(\frac{x}{\ell}\right)^2\delta\ell \tag{18.295}$$

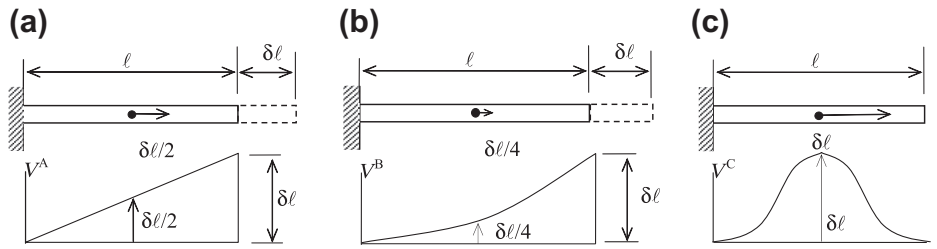


FIGURE 18.29

Midpoint movement with design change $\delta\ell$. (a) $\delta\ell/2$: linear velocity. (b) $\alpha\delta\ell \neq \delta\ell/2$: other than linear velocity. (c) $\delta\ell$: beam length change is zero.

No matter how the velocity field is defined, the z' term is unchanged because the functions z_τ and z are evaluated at the same point. The same is true for $z_{,1}$. The only term that is affected is the velocity field V in the second term of Eq. 18.287. We assume the displacement is measured at midpoint $x = \ell/2$, the design velocities at the midpoint are $\delta\ell/2$ and $\delta\ell/4$, respectively, using V^A and V^B. Hence, the material derivatives of the displacement at the midpoint become

$$\dot{z}^A_{\ell/2} = z' + z_{,1}V_1 = \frac{3q\ell^3}{32EI}\delta\ell + \frac{7q\ell^3}{48EI}\left(\frac{\delta\ell}{2}\right) = \frac{q\ell^3}{6EI}\delta\ell \tag{18.296}$$

and

$$\dot{z}^B_{\ell/2} = z' + z_{,1}V_1 = \frac{3q\ell^3}{32EI}\delta\ell + \frac{7q\ell^3}{48EI}\left(\frac{\delta\ell}{4}\right) = \frac{25q\ell^3}{192EI}\delta\ell \tag{18.297}$$

in which $\dot{z}^A_{\ell/2}$ predicts the displacement at the midpoint of the beam if the movement of the displacement measurement point is determined by velocity field V^A when a beam length is changed by $\delta\ell$. Similarly, $\dot{z}^B_{\ell/2}$ predicts the displacement at the midpoint of the beam if the movement of the displacement measurement point is determined by velocity field V^B. Therefore, in general, domain velocity is not unique, as long as it satisfies the requirements discussed in Section 18.5.4 and adequately captures the design intent.

We now take a look at a design velocity field V^C shown in Figure 18.29(c), which is defined as a quadratic function with zero at both ends and maximum at the midpoint. Mathematically, V^C is defined as

$$V^C = \frac{4x}{\ell^2}(\ell - x)\delta\ell \tag{18.298}$$

In this case, the shape sensitivity coefficient at the tip is zero because $V = 0$ at $x = \ell$. However, at the midpoint, the sensitivity is not zero; instead

$$\dot{z}^C_{\ell/2} = z' + z_{,1}V = 0 + \frac{7q\ell^3}{48EI}(\delta\ell) = \frac{7q\ell^3}{48EI}\delta\ell \tag{18.299}$$

Here, $z' = 0$ because the displacement function is unchanged (length stays constant). The second term of Eq. 18.299 is nonzero, which characterizes the change of displacement due to the change of the measurement point.

With a basic understanding of the concept of shape design, we move on to introduce the shape sensitivity analysis method based on the continuum approach. To simplify the mathematical derivations, we focus only on the beam example.

18.5.7 SHAPE SENSITIVITY ANALYSIS USING THE CONTINUUM APPROACH

We start by stating one of the major material derivative equations that are essential for the continuum approach. Readers are referred to Choi and Kim (2006a) for details and mathematical proof of the equation.

For an integral function defined as

$$\Phi = \int_{\Omega_\tau} f_\tau(\mathbf{x}_\tau)\mathrm{d}\Omega_\tau \tag{18.300}$$

in which f is a scalar function, the material derivative of Φ is given as

$$\dot{\Phi} = \left(\int_{\Omega_\tau} f(\mathbf{x}_\tau)\mathrm{d}\Omega_\tau\right)^{\cdot} = \int_{\Omega} (f'(\mathbf{x}) + \mathrm{div}(f\mathbf{V}))\mathrm{d}\Omega \tag{18.301}$$

in which the divergence is defined as $\mathrm{div}(f\mathbf{V}) = (f\mathrm{V}_1)_{,1} + (f\mathrm{V}_2)_{,2} + (f\mathrm{V}_3)_{,3}$ for 3-D structures.

This equation is applied to the energy bilinear form and load linear form of Eq. 18.277. We rewrite the energy bilinear form as

$$a_{\Omega_\tau}(\mathbf{z}_\tau, \bar{\mathbf{z}}_\tau) = \int_{\Omega_\tau} c(\mathbf{z}_\tau, \bar{\mathbf{z}}_\tau)\mathrm{d}\Omega_\tau \tag{18.302}$$

For a beam structure, $c(\mathbf{z}, \bar{\mathbf{z}}) = EIz_{3,11}\bar{z}_{3,11}$. The material derivative of $a_{\Omega_\tau}(\mathbf{z}_\tau, \bar{\mathbf{z}})$ can be written using Eq. 18.301:

$$\begin{aligned}
\left[\dot{a}_{\Omega_\tau}(\mathbf{z}_\tau, \bar{\mathbf{z}}_\tau)\right] &= \int_{\Omega} [c(\mathbf{z}, \bar{\mathbf{z}})]'\mathrm{d}\Omega + \int_{\Omega} \mathrm{div}[c(\mathbf{z}, \bar{\mathbf{z}})\mathbf{V}]\mathrm{d}\Omega \\
&= \int_{\Omega} [c(\mathbf{z}', \bar{\mathbf{z}}) + c(\mathbf{z}, \bar{\mathbf{z}}')]\mathrm{d}\Omega + \int_{\Omega} \mathrm{div}[c(\mathbf{z}, \bar{\mathbf{z}})\mathbf{V}]\mathrm{d}\Omega \\
&= \int_{\Omega} \left[c\left(\dot{\mathbf{z}} - \boldsymbol{\nabla}\mathbf{z}^T\mathbf{V}, \bar{\mathbf{z}}\right) + c\left(\mathbf{z}, \bar{\mathbf{z}} - \boldsymbol{\nabla}\bar{\mathbf{z}}^{\mathrm{T}}\mathbf{V}\right)\right]\Omega + \int_{\Omega} \mathrm{div}[c(\mathbf{z}, \bar{\mathbf{z}})\mathbf{V}]\mathrm{d}\Omega \\
&= \int_{\Omega} c(\dot{\mathbf{z}}, \bar{\mathbf{z}})\mathrm{d}\Omega - \int_{\Omega} \left[c\left(\boldsymbol{\nabla}\mathbf{z}^{\mathrm{T}}\mathbf{V}, \bar{\mathbf{z}}\right) + c\left(\mathbf{z}, \boldsymbol{\nabla}\bar{\mathbf{z}}^{\mathrm{T}}\mathbf{V}\right)\right]\mathrm{d}\Omega + \int_{\Omega} \mathrm{div}[c(\mathbf{z}, \bar{\mathbf{z}})\mathbf{V}]\mathrm{d}\Omega \\
&= a_{\Omega}(\dot{\mathbf{z}}, \bar{\mathbf{z}}) + a'_{\mathrm{V}}(\mathbf{z}, \bar{\mathbf{z}})
\end{aligned} \tag{18.303}$$

in which we employed the relation $\mathbf{z}' = \dot{\boldsymbol{z}}(\boldsymbol{x}) - \boldsymbol{\nabla}\mathbf{z}^T\mathbf{V}$ and set $\bar{\mathbf{z}}(\boldsymbol{x}) = 0$ because $\dot{\bar{\mathbf{z}}}(\boldsymbol{x}) = \mathbf{0}$ is virtual displacement. We define

$$a_{\Omega}(\dot{\mathbf{z}}, \bar{\mathbf{z}}) \equiv \int_{\Omega} c(\dot{\mathbf{z}}, \bar{\mathbf{z}})\mathrm{d}\Omega \tag{18.304}$$

and

$$a'_{\mathrm{V}}(\mathbf{z},\bar{\mathbf{z}}) \equiv -\int_{\Omega}\left[c\left(\nabla\mathbf{z}^{\mathrm{T}}\mathbf{V},\bar{\mathbf{z}}\right)+c\left(\mathbf{z},\nabla\bar{\mathbf{z}}^{\mathrm{T}}\mathbf{V}\right)\right]\mathrm{d}\Omega+\int_{\Omega}\mathrm{div}[c(\mathbf{z},\bar{\mathbf{z}})\mathbf{V}]\mathrm{d}\Omega \tag{18.305}$$

Repeating the same for the load linear form (we're excluding the traction force for simplicity), we have

$$\begin{aligned}[\dot{\ell_{\Omega_\tau}}(\bar{\mathbf{z}})] &= \int_{\Omega}(\dot{\boldsymbol{f}^{\mathrm{T}}}\bar{\mathbf{z}})\mathrm{d}\Omega = \int_{\Omega}\left(\boldsymbol{f}^{\mathrm{T}}\bar{\mathbf{z}}\right)'\mathrm{d}\Omega+\int_{\Omega}\mathrm{div}\left[\left(\boldsymbol{f}^{\mathrm{T}}\bar{\mathbf{z}}\right)\mathbf{V}\right]\mathrm{d}\Omega\\ &= -\int_{\Omega}\boldsymbol{f}^{\mathrm{T}}\nabla\bar{\mathbf{z}}^{\mathrm{T}}\mathbf{V}\mathrm{d}\Omega+\int_{\Omega}\mathrm{div}\left(\boldsymbol{f}^{\mathrm{T}}\bar{\mathbf{z}}\mathbf{V}\right)\mathrm{d}\Omega\equiv\ell'_{V}(\bar{\mathbf{z}})\end{aligned} \tag{18.306}$$

in which we assume $\boldsymbol{f}'=0$. Hence, equating $[a_{\Omega_\tau}(\mathbf{z}_\tau,\bar{\mathbf{z}}_\tau)]$ and $[\ell_{\Omega_\tau}(\bar{\mathbf{z}}_\tau)]$, we have

$$a_{\Omega}\left(\dot{\mathbf{z}},\bar{\mathbf{z}}\right)=-a'_{V}(\mathbf{z},\bar{\mathbf{z}})+\ell'_{V}(\bar{\mathbf{z}}),\quad\forall\bar{\mathbf{z}}\in S \tag{18.307}$$

Similar to sizing sensitivity analysis, in implementation, the right-hand side of Eq. 18.307 is calculated as a fictitious load, which is employed to solve for $\dot{\mathbf{z}}$. Equation 18.307 represents the direct differentiation method of the continuum approach. For the adjoint variable method of the continuum approach, readers are referred to Choi and Kim (2006a).

18.5.7.1 Shape Sensitivity Analysis for a Cantilever Beam

For one-dimensional beam bending problems, we have $c(\mathbf{z},\bar{\mathbf{z}})=EIz_{3,11}\bar{z}_{3,11}$, $\mathbf{z}=z_3$, $\mathbf{V}^{\mathrm{T}}\mathbf{n}=V$, $\nabla\mathbf{z}=z_{3,1}$, and $\mathrm{div}\,\boldsymbol{f}=f_{,1}$. Hence, from Eqs 18.304, 18.305, and 18.306, we have

$$a_{\Omega}(\dot{\mathbf{z}},\bar{\mathbf{z}})=\int_{0}^{\ell}EI\dot{z}_{3,11}\bar{z}_{3,11}\mathrm{d}x \tag{18.308}$$

$$\begin{aligned}a'_{\mathrm{V}}(\mathbf{z},\bar{\mathbf{z}}) &= -\int_{0}^{\ell}\left[EI\left(z_{3,1}V\right)_{,11}\bar{z}_{3,11}+EIz_{3,11}\left(\bar{z}_{3,1}V\right)_{,11}\right]\mathrm{d}x+\int_{0}^{\ell}\left(EIz_{3,11}\bar{z}_{3,11}V\right)_{,1}\mathrm{d}x\\ &= -3\int_{0}^{\ell}\left(EIz_{3,11}\bar{z}_{3,11}V_{,1}\right)\mathrm{d}x-\int_{0}^{\ell}\left[EI\left(z_{3,11}\bar{z}_{3,1}+z_{3,1}\bar{z}_{3,11}\right)V_{,11}\right]\mathrm{d}x\end{aligned} \tag{18.309}$$

and

$$\begin{aligned}\ell'_{\mathrm{V}}(\bar{\mathbf{z}}) &= -\int_{\Omega}\boldsymbol{f}^{\mathrm{T}}\nabla\bar{\mathbf{z}}^{\mathrm{T}}\mathbf{V}\Omega+\int_{\Omega}\mathrm{div}\left(\boldsymbol{f}^{\mathrm{T}}\bar{\mathbf{z}}V\right)\Omega=-\int_{0}^{\ell}f\,\bar{z}_{3,1}V\mathrm{d}x+\int_{0}^{\ell}\left(f\,\bar{z}_{3}V\right)_{,1}\mathrm{d}x\\ &= \int_{0}^{\ell}\left(f_{,1}V+f\,V_{,1}\right)\bar{z}_{3}\mathrm{d}x\end{aligned} \tag{18.310}$$

Plugging these terms into Eq. 18.307, we have

$$\int_0^\ell EI\dot{z}_{3,11}\bar{z}_{3,11}\mathrm{d}x = 3\int_0^\ell \left(EIz_{3,11}\bar{z}_{3,11}V_{,1}\right)\mathrm{d}x + \int_0^\ell \left[EI\left(z_{3,11}\bar{z}_{3,1} + z_{3,1}\bar{z}_{3,11}\right)V_{,11}\right]\mathrm{d}x + \int_0^\ell \left(f_{,1}V + fV_{,1}\right)\bar{z}_3\mathrm{d}x \tag{18.311}$$

Using finite element shape functions, such as those shown in Eq. 18.143, to discretize the sensitivity equation (Eq. 18.311), we have on the left-hand side

$$\int_0^\ell EI\dot{z}_{3,11}\bar{z}_{3,11}\mathrm{d}x = \bar{\mathbf{Z}}_{\mathrm{g}}^{\mathrm{T}}\mathbf{K}_{\mathrm{g}}\dot{\mathbf{Z}}_{\mathrm{g}} \tag{18.312}$$

where

$$\dot{\mathbf{Z}}_{\mathrm{g}} = \left[\dot{z}_{3_i}\ \ \dot{\theta}_i\ \ \dot{z}_{3j}\ \ \dot{\theta}_j\right]^{\mathrm{T}} \tag{18.313}$$

The right-hand side of Eq. 18.311 can be computed as $\bar{\mathbf{Z}}^{\mathrm{T}}\mathbf{F}_{\mathrm{fic}_{\mathrm{g}}}$, in which $\mathbf{F}_{\mathrm{fic}_{\mathrm{g}}}$ is the global fictitious load vector. By imposing the boundary conditions and removing the virtual displacement $\bar{\mathbf{Z}}$ from both sides, we have

$$\mathbf{K}\dot{\mathbf{Z}} = \mathbf{F}_{\mathrm{fic}} \tag{18.314}$$

which can be solved using the same decomposed stiffness matrix $\mathbf{K}$ as the original structure considering the fictitious load $\mathbf{F}_{\mathrm{fic}}$ as additional load cases. Equation 18.311 represents the direct differentiation method of the continuum-discrete approach for shape sensitivity analysis. We illustrate a few more details in the following example.

EXAMPLE 18.25

Calculate shape sensitivity for the cantilever beam shown in Figure 18.13, assuming a linear design velocity of Eq. 18.294—that is, $V = V^{\mathrm{A}} = \frac{x}{\ell}$ (set $\delta\ell = 1$)—using the continuum-discrete method.

Solution

We first rewrite Eq. 18.311 for the cantilever beam with $f = q$ (hence $f_{,1} = q_{,1} = 0$) and a linear velocity field (hence $V_{,11} = 0$) as

$$\int_0^\ell EI\dot{z}_{3,11}\bar{z}_{3,11}\mathrm{d}x = 3\int_0^\ell \left(EIz_{3,11}\bar{z}_{3,11}V_{,1}\right)\mathrm{d}x + \int_0^\ell qV_{,1}\bar{z}_3\mathrm{d}x \tag{18.315}$$

Using the shape functions of Eq. 18.143, we have

$$z_3 = \mathbf{N}^{\mathrm{T}}\mathbf{Z} = \begin{bmatrix} N_3 & N_4 \end{bmatrix}\begin{bmatrix} z_{3_j} \\ \theta_j \end{bmatrix} = \begin{bmatrix} \frac{3x^2}{\ell^2} - \frac{2x^3}{\ell^3} & -\frac{x^2}{\ell} + \frac{x^3}{\ell^2} \end{bmatrix}\begin{bmatrix} \frac{q\ell^4}{8EI} \\ \frac{q\ell^3}{6EI} \end{bmatrix} = \frac{q\ell}{24EI}\left(5x^2\ell - 2x^3\right) \tag{18.316}$$

$$z_{3,1} = \left(\mathbf{N}^{\mathrm{T}}\mathbf{Z}\right)_{,1} = \begin{bmatrix} N_{3,1} & N_{4,1} \end{bmatrix}\begin{bmatrix} z_{3_j} \\ \theta_j \end{bmatrix} = \begin{bmatrix} \frac{6x}{\ell^2} - \frac{6x^2}{\ell^3}f - \frac{2x}{\ell} + \frac{3x^2}{\ell^2} \end{bmatrix}\begin{bmatrix} \frac{q\ell^4}{8EI} \\ \frac{q\ell^3}{6EI} \end{bmatrix} = \frac{q\ell}{12EI}\left(5x\ell - 3x^2\right) \tag{18.317}$$

Continued

EXAMPLE 18.25—cont'd

$$z_{3,11} = \left(\mathbf{N}^{\mathrm{T}}\mathbf{Z}\right)_{,11} = \begin{bmatrix} N_{3,11} & N_{4,11} \end{bmatrix} \begin{bmatrix} z_{3_j} \\ \theta_j \end{bmatrix} = \begin{bmatrix} \frac{6}{\ell^2} - \frac{12x}{\ell^3} f - \frac{2}{\ell} + \frac{6x}{\ell^2} \end{bmatrix} \begin{bmatrix} \frac{q\ell^4}{8EI} \\ \\ \frac{q\ell^3}{6EI} \end{bmatrix} = \frac{q\ell}{12EI}(5\ell - 6x) \tag{18.318}$$

Bringing Eqs 18.316 to 18.318 to the right-hand side of Eq. 18.315, we have

$$\int_0^\ell EIz_{3,11}\bar{z}_{3,11}V_{,1}\mathrm{d}x = \overline{\mathbf{Z}}^{\mathrm{T}} \int_0^\ell \begin{bmatrix} \frac{6}{\ell^2} - \frac{12x}{\ell^3} \\ \\ \frac{-2}{\ell} + \frac{6x}{\ell^2} \end{bmatrix} \left[\frac{q\ell}{12}(5\ell - 6x)\right] \frac{1}{\ell}\mathrm{d}x = \overline{\mathbf{Z}}^{\mathrm{T}} \frac{q}{12} \int_0^\ell \begin{bmatrix} \frac{6}{\ell^2} - \frac{12x}{\ell^3} \\ \\ \frac{-2}{\ell} + \frac{6x}{\ell^2} \end{bmatrix} (5\ell - 6x)\mathrm{d}x = \overline{\mathbf{Z}}^{\mathrm{T}} \begin{bmatrix} \frac{q}{2} \\ \\ \frac{-q\ell}{12} \end{bmatrix} \tag{18.319}$$

and

$$\int_0^\ell q\bar{z}_3 V_{,1}\mathrm{d}x = \overline{\mathbf{Z}}^{\mathrm{T}} \int_0^\ell q \begin{bmatrix} \frac{3x^2}{\ell^2} - \frac{2x^3}{\ell^3} \\ \\ \frac{-x^2}{\ell} + \frac{x^3}{\ell^2} \end{bmatrix} \frac{1}{\ell}\mathrm{d}x = \overline{\mathbf{Z}}^{\mathrm{T}} \frac{q}{\ell} \int_0^\ell \begin{bmatrix} \frac{3x^2}{\ell^2} - \frac{2x^3}{\ell^3} \\ \\ \frac{-x^2}{\ell} + \frac{x^3}{\ell^2} \end{bmatrix} \mathrm{d}x = \overline{\mathbf{Z}}^{\mathrm{T}} \begin{bmatrix} \frac{q}{2} \\ \\ \frac{-q\ell}{12} \end{bmatrix} \tag{18.320}$$

Bringing Eqs 18.319 and 18.320 to 18.315, we have

$$\overline{\mathbf{Z}}^{\mathrm{T}} \left\{ 3 \begin{bmatrix} \frac{q}{2} \\ \\ \frac{-q\ell}{12} \end{bmatrix} + \begin{bmatrix} \frac{q}{2} \\ \\ \frac{-q\ell}{12} \end{bmatrix} \right\} = \overline{\mathbf{Z}}^{\mathrm{T}} \begin{bmatrix} 2q \\ \\ \frac{-q\ell}{3} \end{bmatrix} = \overline{\mathbf{Z}}^{\mathrm{T}} \mathbf{F}_{\mathrm{fic}} \tag{18.321}$$

By solving the sensitivity vector in the same way as solving $\mathbf{Z}$ of the structure (see 18.149), we have

$$\begin{bmatrix} \dot{z}_{3j} \\ \dot{\theta}_j \end{bmatrix} = \mathbf{K}^{-1}\mathbf{F}_{\mathrm{fic}} = \frac{\ell}{12EI} \begin{bmatrix} 4\ell^2 & 6\ell \\ 6\ell & 12 \end{bmatrix} \begin{bmatrix} 2q \\ \\ \frac{-q\ell}{3} \end{bmatrix} = \begin{bmatrix} \frac{q\ell^3}{2EI} \\ \\ \frac{2q\ell^2}{3EI} \end{bmatrix} \tag{18.322}$$

18.6 TOPOLOGY OPTIMIZATION

Sizing optimization varies the sizes of the structural elements, such as the diameter of a bar or beam, as shown in Figure 18.30(a), or the thickness of a sheet metal. In sizing optimization, the shape of the structure is known and unchanged. For shape optimization, the topology of geometric features in the structure, such as the number of holes, is known. The optimal shape is confined to the topology of the initial structural geometry, as illustrated in Figure 18.30(b). No additional holes can be created during the shape optimization process. For both sizing and shape optimization, the topology of the structural geometry is unchanged in the design process.

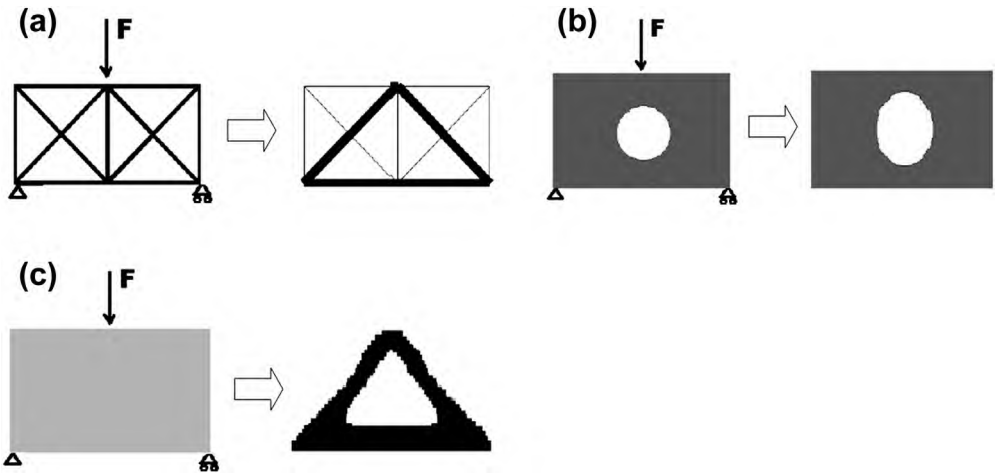


FIGURE 18.30

Different types of structural optimization. (a) Sizing optimization. (b) Domain shape optimization. (c) Topology optimization.

Another important type of optimization problem is topology optimization, which solves the basic engineering problem of distributing a prescribed amount of material in a design space. Topology optimization is sometimes referred to as layout optimization. In general, the goal of topology optimization is to find the best use of material for a structural body that is subject to either a single load or multiple load distributions. The best use of material in the case of topology optimization represents a maximum stiffness (or minimum compliance), maximum buckling load, or maximum first vibration frequency design. With topology optimization, the resulting shape or topology is not known; the number of holes, structural members (for frame structures), etc. are not decided upon, as illustrated in Figure 18.30(c).

In topology optimization, we start from a given design domain and proceed to find an optimum distribution of material and voids. In general, a design domain is discretized by using the FEM into discrete mesh. Individual finite elements in the discretized design domain are then evaluated to determine if they are retained or removed from the structure using optimization methods. This results in a so-called 0–1 problem—the elements either exist or do not.

The two main solution strategies for solving the topology optimization problem are the density method and the homogenization method. In this section, we briefly discuss the density method. Those interested in learning the homogenization method are referred to Bendsøe and Sigmund (2003).

18.6.1 BASIC CONCEPT AND PROBLEM FORMULATION

In general, the objective of a topology optimization problem is either to minimize the compliance, which is equivalent to maximizing the stiffness, maximize the lowest frequency, or maximize the lowest buckling load. A volume constraint is imposed to limit material usage. The optimization problem can be formulated as

$$\text{Minimize}: \quad \phi(c(\mathbf{x})) \tag{18.323a}$$

$$\text{Subject to}: \int_{\Omega} c(\mathbf{x})\rho_0 d\Omega \le m_0 \tag{18.323b}$$

$$\psi_i - \psi_i^u \le 0, \quad i = 1, m \tag{18.323c}$$

$$0 \le c(\mathbf{x}) \le 1 \tag{18.323d}$$

where ϕ is a structural performance measure to be minimized (or maximized), such as compliance; ψ_i is the ith structural performance constraint, such as displacement; m is the total number of constraints; Ω is the structural domain; m_0 is the mass limit; $c(\mathbf{x})$ is the design variable; and ρ_0 is material mass density. In implementation, the design variable $c(\mathbf{x})$ is assigned to individual finite elements; therefore, $c(\mathbf{x}) = c_i$, $i = 1$, NE (number of elements). In each element, the density becomes $\rho_i = c_i\ \rho_0$, and Eq. 18.323b becomes

$$\int_{\Omega} c(\mathbf{x})\rho_0 d\Omega = \sum_{i=1}^{\text{NE}} c_i \rho_0 V_i \le m_0 \tag{18.324}$$

where V_i is the volume (or area for planar problems) of the ith finite element. Also, Eq. 18.323d becomes

$$0 \le c_i \le 1, \quad i = 1, \quad \text{NE} \tag{18.325}$$

It is desired that the solution of topology optimization only consists of solid or empty elements—that is, $c_i = 0$ or 1. One popular method to suppress or penalize intermediate densities is by letting the stiffness of the material be expressed as

$$E = c_i^p E_0, \quad i = 1, \quad \text{NE} \tag{18.326}$$

where E_0 is the modulus of elasticity of the material and p is a penalization factor that is greater than zero, typically 2 to 5 in implementation. For $p > 1$, local stiffness for values of $c_i < 1$ is lowered, as illustrated in Figure 18.31, thus making it "uneconomical" to have intermediate densities in the

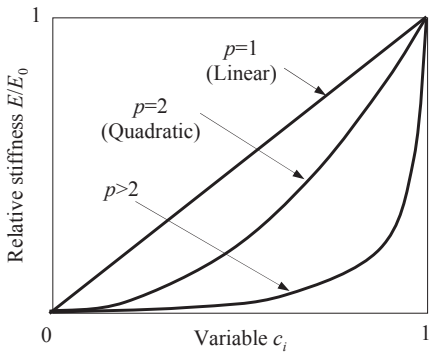


FIGURE 18.31

Relative stiffness as a function of variable c_i with different penalization factors p (Eschenauer and Olhoff, 2001).

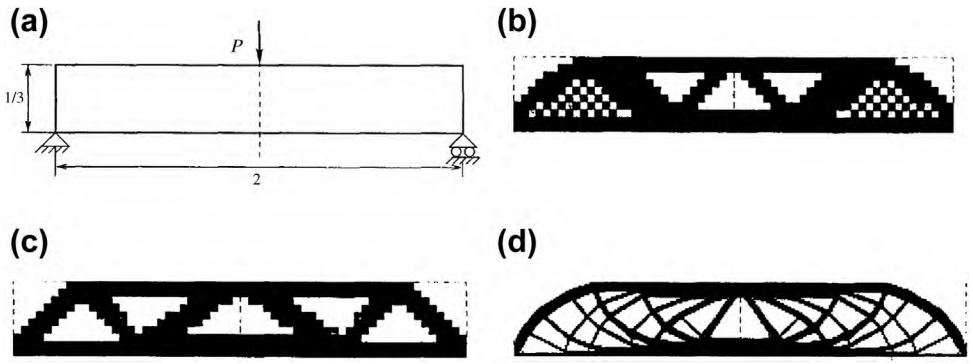


FIGURE 18.32

Numerical instabilities in topology optimization. (a) Design problem. (b) Example of checkerboards. (c) Solution for 600-element discretization. (d) Solution for 5400-element discretization (Sigmund and Petersson, 1998).

optimal design. In the literature, the density method together with this penalization is often called the SIMP (solid isotropic microstructures with penalization) method (Rozvany, 2000).

Although the density method is straightforward to implement for topology optimization, one must be aware of numerical instabilities. These can manifest as checkerboard, mesh dependence, and local minimum. More detailed discussion and remedies to treat these instabilities can be found in Sigmund and Petersson (1998).

Topology optimization often converges to a design that contains a checkerboard pattern—that is, the alternately solid and void elements, as shown in Figure 18.32(b). This is a typical result in topology optimization using finite element methods.

The quality of topology optimization result is dependent on the discretization of the finite element model, the so-called mesh-dependence problem. Figure 18.32(c) shows the topology optimization result for the discretization using 600 elements, and Figure 18.32(d) shows the result using 5400 elements for the same physical problem. The result in Figure 18.32(d) is much more detailed than that in Figure 18.32(c); however, the two topologies are different in nature.

Different solutions to the same discretized problem are observed when choosing different parameters, such as finite element type (triangular vs quadratic elements), optimization algorithms, convergence criteria, and so forth. These numerical instabilities are all related to whether the continuous constrained optimization problems are well posed or not.

18.6.2 TWO-DIMENSIONAL CANTILEVER BEAM EXAMPLE

A cantilever beam with a fixed left end and a vertical load applied at the midpoint of the free end, as shown in Figure 18.33(a), is used to illustrate the density method for topology optimization. The material properties are modulus of elasticity $E = 2.07 \times 10^5$ psi and Poisson's ratio $\nu = 0.3$.

The finite element model contains 32×20 mesh with four-node quad elements. There are 640 elements in the model and the density of each element is selected as the design variable. The

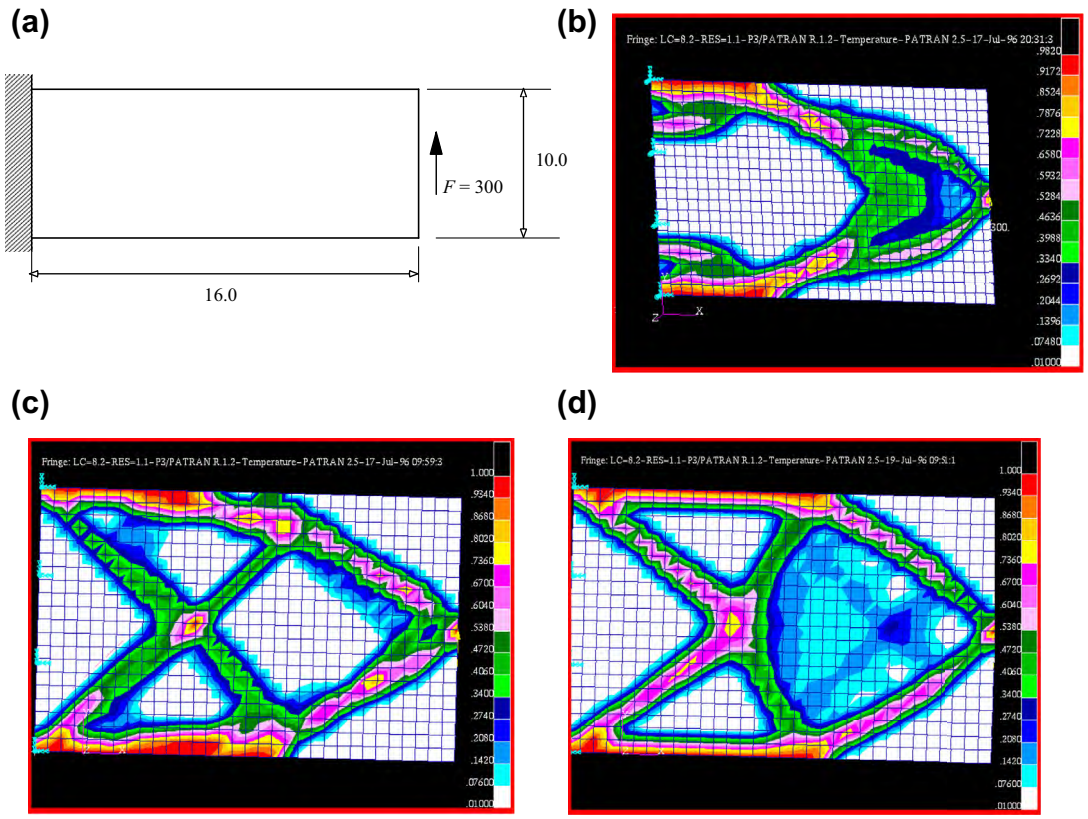


FIGURE 18.33

The two-dimensional topology optimization example. (a) Analysis model (dimensions are in inches and force is in pounds). (b) Optimal topology design using modified feasible direction. (c) Optimal topology design using sequential linear programming. (d) Optimal topology design using sequential quadratic programming.

design problem is to minimize the compliance with the area constraint of 25% imposed on the domain. Figure 18.33(b) shows the material distribution using the modified feasible direction after 10 iterations. Figure 18.33(c) shows the material distribution using the sequential linear programming after 36 iterations. Figure 18.33(d) shows the material distribution using the sequential quadratic programming after 28 iterations. They converge to slightly different topologies.

18.7 CASE STUDY

Topology optimization has drawn significant attention in the recent development of structural optimization. This method has been proven very effective in determining the initial geometric shape or layout for structural design. The major drawback of the method, however, is that the topology optimization

leads to a nonsmooth structural geometry, while most of the engineering applications require a smooth geometric shape, especially for manufacturing. On the other hand, shape optimization starts with a smooth geometric model that can be manufactured much more easily. However, the optimal shape is confined to the topology of the initial structural geometry. No additional holes can be created during the shape optimization process. It is desirable to integrate topology and shape optimizations in support of structural design by taking advantage of both methods. In this case study, we present structural optimization for a tracked vehicle roadarm by integrating topology optimization with shape optimization.

18.7.1 THE TRACKED VEHICLE ROADARM

A tracked vehicle roadarm example (Tang and Chang, 2001) shown in Chapter 2, Figure 2.34(a) is presented to illustrate the design process. The roadarm transfers forces exerted on the roadwheel of the tracked vehicle to the torsion bar in the vehicle suspension system while the vehicle is maneuvering.

18.7.2 TOPOLOGY OPTIMIZATION

Topology optimization is carried out to obtain an optimal structural layout, in this case using OptiStruct (www.altairhyperworks.com). At the beginning of the topology optimization, a bulk shape is assumed (Figure 2.34(b)). The end of the torsion bar is fully constrained and an axial force of 2.15×10^4 lb. is applied at the wheel shaft, as shown in Figure 2.34(b). The objective of the topology optimization is to minimize the structural compliance subject to a 45% mass reduction. The original finite element model and its topologically optimized layout are shown in Figures 2.34(b) and (c), respectively.

18.7.3 BOUNDARY SMOOTHING

In this roadarm example, cubic B-spline curves and surfaces, as discussed in Chapter 2, are utilized to approximate and smooth the boundary points of the irregular structural layout. During this process, the control points, which govern the geometric shape of the B-spline surfaces, are acquired. Then, through the CAD API, these control points are brought into the SolidWorks environment for solid model construction. These imported control points also parameterize the boundary surfaces of the reconstructed solid model and later serve as design variables for shape optimization.

As an example, the geometric points of five representative sections of the roadarm (Figure 2.34(c)) are selected and fitted with B-spline curves (Step 2 of Figure 2.35). Following the surface skinning technique (Tang and Chang, 2001), an outer polygon surface formed by 6×5 control points and the enclosed B-spline surface are created (Step 3a of Figure 2.35). Similarly, an inner B-spline surface (4×3 control points) that represents the hole in the roadarm is created (Step 3b of Figure 2.35). Note that the B-spline surface constructed is C^2-continuous in both u- and w-parametric directions because cubic basis functions are employed. The control points and basis functions of the B-spline surface can be imported into CAD tools, in this case SolidWorks, to support solid modeling and shape optimization.

In SolidWorks, the outer and inner solid models are created by filling out the cavities enclosed by the outer and inner B-spline surfaces, respectively. The final solid model is obtained by subtracting the

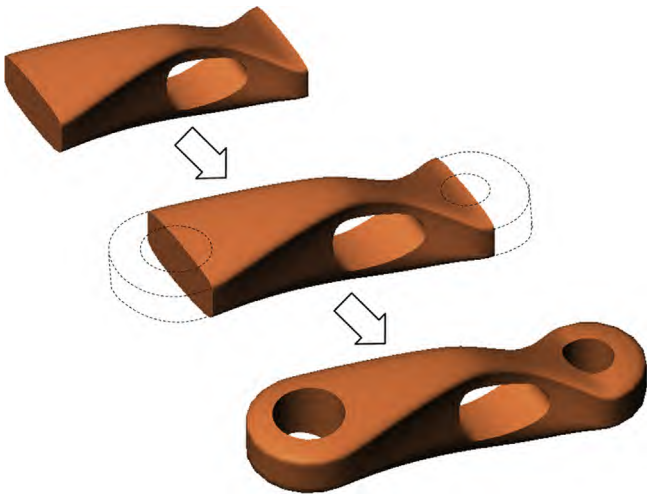


FIGURE 18.34

Reconstruction of roadarm solid model.

inner solid from the outer one and uniting the subtracted solid model with two end half cylinders, as illustrated in Figure 18.34. Once the solid model is constructed, finite element mesh can be created using, for example, automatic mesh generation in CAD or FEA.

18.7.4 SHAPE PARAMETERIZATION AND DESIGN VELOCITY FIELD COMPUTATION

The movements of the selected control points of the B-spline surfaces in the x_1, x_2, and x_3 directions can be defined as design variables for shape optimization. The boundary design velocity field of a node N defined by (u_N, w_N) parameters of the B-spline surface can be computed using the methods discussed in Section 18.5.4.

Note that the parametric coordinates (u_N, w_N) of a given finite element boundary node N are determined by CAD via its API function. For example, in SolidWorks, the API function GetClosestPointsOn (x_1, x_2, x_3), where (x_1, x_2, x_3) are the Cartesian coordinates of the finite element node, can be used to retrieve the parametric coordinates (u_N, w_N) of the node (in addition to those of Pro/ENGINEER mentioned in Section 18.5.4).

According to the design variables shown in Figure 18.35, the outer and inner B-spline surfaces will vary due to design changes. When the control points shared by the outer and inner B-spline surfaces move, the material points on the intersection edges can move following either the outer or inner surface. Such movements are not unique. To alleviate the problem, the boundary velocity field of both the outer and inner surfaces is determined by the outer surface only. The parametric locations of the finite element nodes of the intersection edges on the outer B-spline surface are first identified along with the other boundary nodes. The boundary velocity field on the outer surface, including the intersection edges, can then be computed. The velocity field on the inner cylindrical surface is determined by the linear interpolation of the edge velocity toward the interior of the surface (along the x_1-direction). This can be easily achieved for the roadarm example because the inner surface is a

cylindrical surface. Once the boundary velocity field is computed, the domain velocity can be calculated using the boundary displacement method discussed in Section 18.5.4.

18.7.5 SHAPE DESIGN OPTIMIZATION

In shape optimization, the objective function is structural mass, and constraint functions are the structural compliance measure obtained from topology optimization and stress measures. Note that the stress upper bound is defined as 37.5 ksi, and the material is SAE 1045 carbon steel with a yield strength of 45 ksi. Hence, a safety factor of 1.2 is employed for the design.

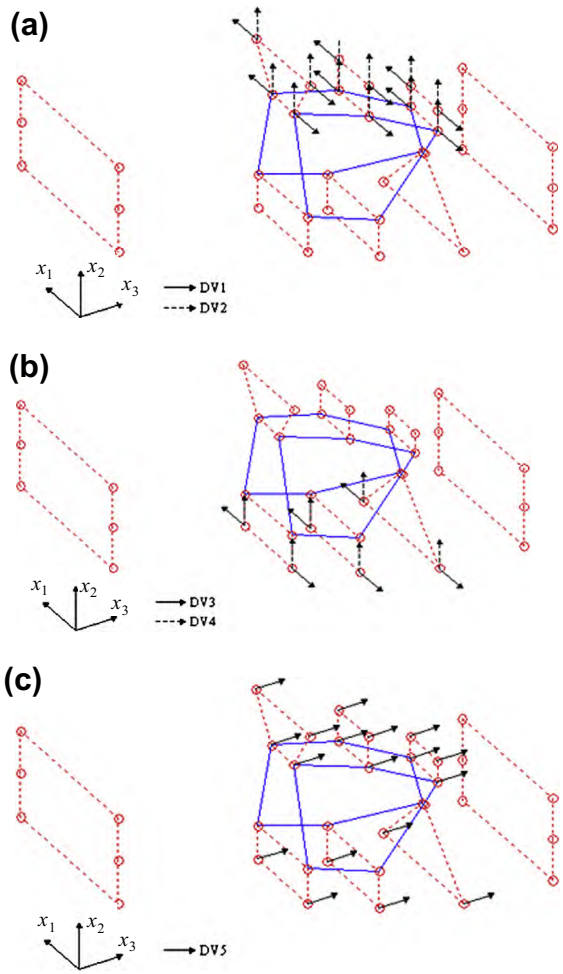


FIGURE 18.35

Shape design parameterization. (a) Design variables DV1 and DV2. (b) Design variables DV3 and DV4. (c) Design variable DV5.

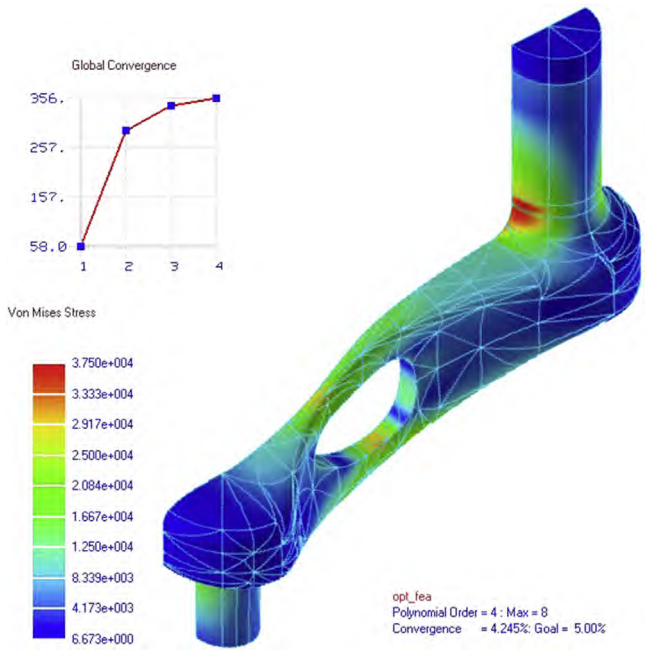


FIGURE 18.36

von Mises stress of the optimal roadarm.

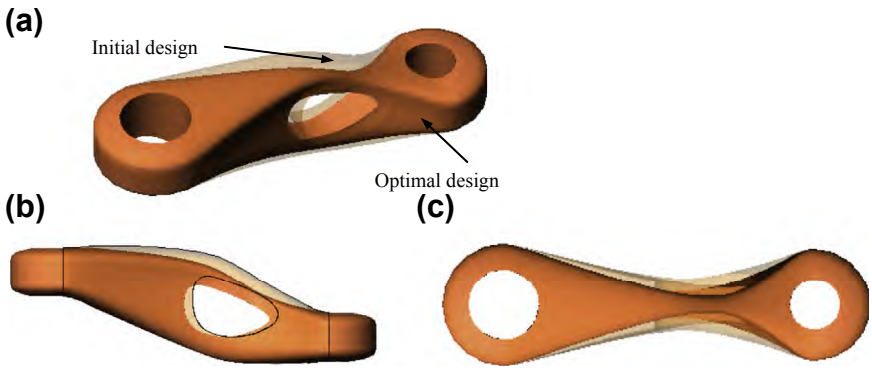


FIGURE 18.37

Comparison of the 3-D roadarm at initial and optimal designs. (a) Isometric view (light color: initial design, dark color: optimal design). (b) Side view. (c) Top view.

The optimal shape was obtained in five design iterations. The von Mises stress distribution of the optimal design is shown in Figure 18.36. The high stress areas are distributed around the upper and lower branches of the roadarm. The highest stress is found located at the top surface of the lower branch with a value of 3.69×10^4 psi, which is less than the allowable stress—that is, material yield

strength divided by the safety factor. The roadarm shape variation between the initial and optimal designs is shown in Figure 18.37. The total mass and compliance reductions are 47% and 81%, respectively, from the initial shape.

18.8 SUMMARY

Efficient and accurate sensitivity analysis is essential to the success of gradient-based optimization, especially for large-scale applications that require substantial computational effort for function evaluations. We discussed three major approaches for sensitivity analysis—the brute-force finite difference, discrete, and continuum methods. We pointed out the pros and cons of individual approaches. In practical implementation, the semi-analytical and continuum-discrete methods are general and efficient. In addition, considering accuracy, the continuum-discrete method is, in our opinion, the best approach for support of DSA in general and large-scale structural problems. However, it requires a more in-depth study for the reader to become proficient in implementing the method and integrating it with commercial FEA software. We hope the discussions provided in this chapter offer a gateway for those who intend to get into a more in-depth study in the sensitivity analysis area. We also briefly discussed topology optimization, which has gained lots of attention in recent years in support of structural layout design. For those who are interested in pursuing graduate study, topology optimization and relevant subjects can be an interesting thesis topic.

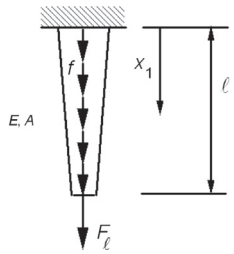
We hope that this chapter has helped readers gain a general understanding of the concept and computation methods for DSA, and be able to implement some of the methods for their own applications. We hope, at this point, that software offering sensitivity analysis capabilities for gradient-based optimization, such as MSC/NASTRAN, is no longer seen as a black box.

QUESTIONS AND EXERCISES

18.1. The governing equation of the bar shown below is given as

$$-\left(EAz_{1,1}\right)_{,1} = f,\ x_1 \in [0, \ell]$$

with $z_1(0) = 0, f(\ell) = F_\ell$



where

- f: body force, $f = \rho g A$
- E: modulus of elasticity
- A: cross-sectional area of the bar defined as

$$A(x_1) = A_0\left(1 - \frac{x_1}{\ell}\right) + \frac{x_1}{\ell}A_\ell$$

A_0 and A_ℓ are the areas at $x = 0$ and $x = \ell$, respectively

- ρ: mass density
- g: gravitational acceleration
- ℓ: length of the bar
- F_ℓ: point force applied at the bottom end of the bar

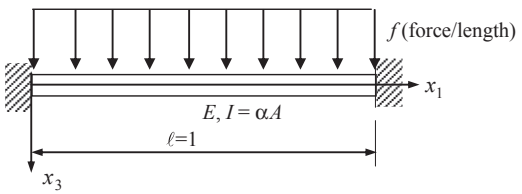
a. Define the space $\overline{S}$ for the solution z_1 of the above partial differential equation, similar to that of Eq. 18.7.
b. Derive the energy bilinear form and load linear form for the bar.
c. Show that the energy bilinear form is bilinear and load linear form is linear.
d. Define the space of kinematically admissible virtual displacement S (as in Eq. 18.12) for the bar structure.
e. State the principle of virtual work for the bar structure. Identify trial and test functions, as well as trial and test function spaces.
f. Solve for z_1 of the bar as a function of x_1.

18.2. Continue with Problem 18.1.
a. Discretize the bar by using two equal-length finite elements of the same shape function shown in Eq. 18.17. Formulate the finite element matrix equations.
b. Solve the matrix equations for the nodal displacements and use the same shape function to write the solution z_1 as functions of x_1 for individual finite elements.
c. Compare the finite element solution with the exact solution obtained in Problem 18.1(f), and comment on the differences between these two solutions.

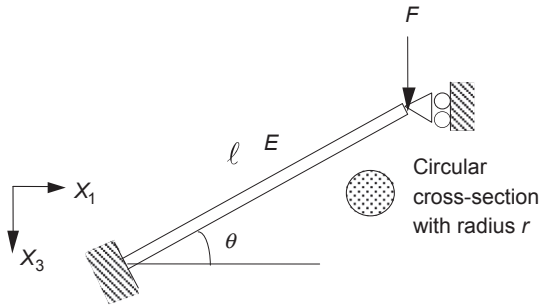
18.3. Continue with Problem 18.2. Assume the modulus of elasticity E and cross-sectional area A_2 of the second element of the bar as design variables.
a. Calculate sensitivity of the displacement at the bottom end of the bar for the two design variables using the direct differentiation method of the discrete-analytical approach.
b. Repeat (a) by using the adjoint variable method of the discrete-analytical approach. Show the adjoint loads on the bar structure.

18.4. Continue with Problem 18.3.
a. Calculate the sensitivity of the stress at the top end of the bar for the two design variables using the direct differentiation method of the continuum-discrete approach.
b. Repeat (a) by using the adjoint variable method of the continuum-discrete approach. Show the adjoint loads on the bar structure.

18.5. For the following clamped–clamped beam, calculate the displacement sensitivity for both modulus E and area A using the direct differential method of the continuum-analytical approach. Note that no shape function can be used.



18.6. Answer the following questions for the beam structure shown below.



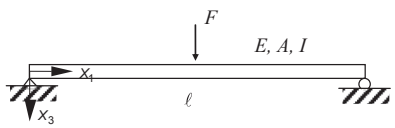
- **a.** Compute and identify the maximum stress along the beam (bending plus axial) using the FEM (one finite element, linear functions for truss effect, and cubic shape functions for bending effect).
- **b.** Define the maximum stress obtained in (a) as the performance measure and radius r as the design variable. Compute the adjoint load using the continuum-discrete approach, and sketch the adjoint load on the beam.
- **c.** Compute the adjoint load for the same problem using the discrete approach.
- **d.** Compute stress sensitivity with respect to the radius defined in (b) using the adjoint variable method of the continuum-discrete approach.
- **e.** Explain the meaning of the term $\int_0^\ell g_{,b}\delta b \mathrm{d}x$, as seen in Eq. 18.173, in this problem. Why is it nonzero? What does it mean if we force it to be zero?
- **f.** Compute stress sensitivity with respect to the radius design variable defined in (b) using the direct differentiation method of the discrete-analytical approach, and sketch the fictitious loads on the beam.

18.7. Four control points on the x–y plane are given as follows:
$\mathbf{P}_0 = [0, 0]$, $\mathbf{P}_1 = [1, 4]$, $\mathbf{P}_2 = [2, -5]$, $\mathbf{P}_3 = [3, 8]$.

- **a.** Construct a Bézier curve enclosed by the control polygon formed by the four given points.
- **b.** If the control point $\mathbf{P}_3$ is moved upward by two units (i.e., $\delta P_{3y} = 2$), calculate the change in curve $\delta\mathbf{P}(u)$, and calculate the parametric equation of the perturbed Bézier curve.
- **c.** For a node with parametric coordinate $u = 0.5$, calculate its velocity field.

18.8. Continue with Problem 18.7. Assume the Bézier curve is edge 4 of a Coons patch. Calculate the velocity for a node inside the patch with $(u, w) = (0.5, 0.5)$.

18.9. Consider the simple support beam loaded with a point load F in the middle, as shown below.

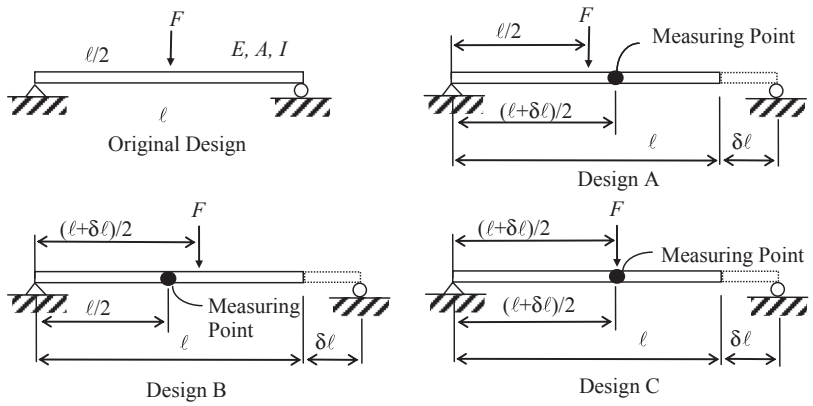


- **a.** Compute the sensitivity of displacement z_3 at the middle of the beam with respect to its length ℓ, assuming $\partial F/\partial \ell = 0$ and that the measuring point is changing with $\delta\ell$.
- **b.** What is the meaning of $\dot{z}_3$ in (a), what is $\dot{z}_3$ predicting?

c. What are the boundary and loading conditions of the perturbed design in (a)?
d. What is $\partial z_3/\partial \ell$, if F does not change with $\delta\ell$?
e. What is $\partial z_3/\partial \ell$, if z_3 is measured at the fixed point?

18.10. This problem is a continuation of Problem 18.9.

A simple support beam loaded with a point load F in the middle is shown below. Answer the following questions for designs A, B, and C shown below:



a. Is the sensitivity $\partial z_3/\partial \ell$ identical for all three designs? Why or why not?
b. In (a), $\partial z_3/\partial \ell = \dot{z} = z' + \nabla z^T V$, calculate z' for all three designs.
c. Calculate sensitivity $\dot{z}$ at $x = \ell/4$ for Design B, and at $x = (\ell + \delta\ell)/4$ for designs A and C.

REFERENCES

Abraham, F.F., Brodbeck, D., Rafey, R.A., Rudge, W.E., 1994. Instability dynamics of fracture: a computer simulation investigation. Physical Review Letters 73 (2), 272–275.

Atkinson, K., 1989. An Introduction to Numerical Analysis, second ed. Wiley, New York.

Bendsøe, M.P., Sigmund, O., 2003. Topology Optimization, Theory, Methods, and Applications, second ed. Springer, New York.

Buehler, M.J., 2008. Atomistic Modeling of Materials Failure. Springer, New York.

Chang, K.H., Joo, S.H., 2006. Design parameterization and tool integration for CAD-based mechanism optimization. Advances in Engineering Software 37, 779–796.

Choi, K.K., Chang, K.H., 1994. A study of design velocity field computation for shape optimal design. Finite Elements in Analysis and Design 15, 317–341.

Choi, K.K., Kim, N.H., 2006a. Structural Sensitivity Analysis and Optimization 1: Linear Systems (Mechanical Engineering Series). Springer, New York.

Choi, K.K., Kim, N.H., 2006b. Structural Sensitivity Analysis and Optimization 2: Nonlinear Systems and Applications (Mechanical Engineering Series). Springer, New York.

Edke, M., Chang, K.H., 2011. Shape optimization for 2-D mixed mode fractures using extended FEM (XFEM) and level set method (LSM). Structural and Multidisciplinary Optimization 44 (2), 165–181.

Eschenauer, H.A., Olhoff, N., 2001. Topology optimization of continuum structures: a review. Applied Mechanics Reviews 54 (4), 331–390.

Hardee, E., Chang, K.H., Tu, J., Choi, K.K., Grindeanu, I., Yu, X., 1999. CAD-based shape design sensitivity analysis and optimization. Advances in Engineering Software 30 (3), 153–175.

Haug, E.J., Choi, K.K., Komkov, V., 1986. Design Sensitivity Analysis of Structural Systems. Academic Press, New York.

Jones, J.E., 1924. On the determination of molecular fields. II. From the equation of state of a gas. Proceedings of the Royal Society of London A 106 (738), 463–477.

Kelchner, C.L., Plimton, S.J., Hamilton, J.C., 1998. Dislocation nucleation and defect structure during surface indentation. Physical Review B 58 (17), 11085–11088.

Mortenson, M.E., 2006. Geometric Modeling, third ed. Industrial Press.

Park, H.S., Karpov, E.G., Klein, P.A., Liu, W.K., 2005. Three-dimensional bridging scale analysis of dynamic fracture. Journal of Computational Physics 207, 588–609.

Rozvany, G.I.N., 2000. Aims, scope, methods, history and unified terminology of computer aided topology optimization in structural mechanics. Structural Multidisciplinary Optimization 21, 90–108.

Sigmund, O., Petersson, J., 1998. Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. Structural Optimization 16, 68–75.

Tang, P.S., Chang, K.H., 2001. Integration of topology and shape optimizations for design of structural components. Journal of Structural Optimization 22 (1), 65–82.

Twu, S., Choi, K.K., 1993. Configuration design sensitivity analysis of built-up structures part II: numerical method. International Journal for Numerical Methods in Engineering 36, 4201–4222.

Wagner, G.J., Liu, W.K., 2003. Coupling of atomistic and continuum simulations using a bridging scale decomposition. Journal of Computational Physics 190, 249–274.

CHAPTER

MULTIOBJECTIVE OPTIMIZATION AND ADVANCED TOPICS 19

CHAPTER OUTLINE

19.1 Introduction 1107
19.2 Basic Concept 1110
19.2.1 Criterion Space and Design Space 1110
19.2.2 Pareto Optimality 1111
19.2.3 Generation of Pareto Optimal Set 1113
19.3 Solution Techniques 1116
19.3.1 Normalization of Objective Functions 1116
19.3.2 Methods with a Priori Articulation of Preferences 1117
19.3.2.1 Weighted-Sum Method 1117
19.3.2.2 Weighted Min–Max Method 1122
19.3.2.3 Lexicographic Method 1124
19.3.3 Methods with A Posteriori Articulation of Preferences 1125
19.3.3.1 Normal Boundary Intersection Method 1126
19.3.4 Methods with No Articulation of Preference 1130
19.3.5 Multiobjective Genetic Algorithms* 1131
19.3.5.1 Pareto-Based Approaches 1132
19.3.5.2 Nondominated Sorting Genetic Algorithm II 1133
19.3.5.3 Sample MATLAB Implementation 1137
19.4 Decision-Based Design 1141
19.4.1 Utility Theory as a Design Tool: Cantilever Beam Example 1141
19.4.2 Game Theory as a Design Tool: Pressure Vessel Example 1144
19.5 Software Tools 1145
19.5.1 Academic Codes 1145
19.5.2 Commercial Tools 1147
19.6 Advanced Topics* 1151
19.6.1 Reliability-Based Design Optimization 1151
19.6.1.1 Failure Probability 1152
19.6.1.2 RBDO Problem Formulation 1152
19.6.1.3 RBDO for a Tracked-Vehicle Roadarm 1155
19.6.2 Design Optimization for Structural Performance and Manufacturing Cost 1158
19.6.2.1 Design Problem Definition and Optimization Process 1159
19.6.2.2 Manufacturing Cost Model 1160
19.6.2.3 Virtual Manufacturing 1161
19.6.2.4 Design Sensitivity Analysis 1162

e-Design. http://dx.doi.org/10.1016/B978-0-12-382038-9.00019-3
Copyright © 2015 Elsevier Inc. All rights reserved.

19.6.2.5 Software Implementation 1163
19.6.2.6 Aircraft Torque Tube Example 1165
19.7 Summary **1170**
Questions and Exercises **1170**
References **1172**

Multiobjective optimization (also known as multiobjective programming, vector optimization, multicriteria optimization, multiattribute optimization, or Pareto optimization) is an area of multiple-criteria decision making, concerning mathematical optimization problems involving more than one objective function to be optimized simultaneously. Multiobjective optimization has been applied to many fields of science and engineering, where optimal decisions need to be taken in the presence of trade-offs between two or more objectives that may be in conflict. Indeed, in many practical engineering applications, designers are making decisions between conflict objectives, such as maximizing performance while minimizing fuel consumption and emission of pollutants of a vehicle. In these cases, a multiobjective optimization study should be performed, which provides multiple solutions representing the trade-offs among the objective functions.

Even for a trivial multiobjective optimization problem, it is unlikely that a single solution that simultaneously optimizes each objective exists. In many cases, the objective functions are said to be in conflict, and there exist (a possibly infinite number of) Pareto optimal solutions. A solution is called nondominated if none of the objective functions can be improved in value without degrading some of the other objective function values. Such solutions are called Pareto optimal. Without additional preference information, all Pareto optimal solutions are considered equally good.

Francis Y. Edgeworth (1845–1926) and Vilfredo Pareto (1848–1923) are credited with first introducing the concept of noninferiority in the context of economics (de Weck, 2004). Since then, multiobjective optimization has permeated engineering and design. The translation of Pareto's work into English in 1971 spurred the development of multiobjective methods in applied mathematics and engineering. The growth of this field manifested itself particularly strongly in the United States, with pioneering contributions made by Stadler (1979) and Steuer (1985), among many others. Another major development, particularly in the theoretical aspects of multiobjective optimization, can be found in Japan (Sawaragi et al., 1985). Over the last three decades, the applications of multiobjective optimization have grown steadily in many areas of engineering and design.

Researchers have studied multiobjective optimization problems from different viewpoints; thus, there exist different solution philosophies and goals when setting and solving them. The goal may be to find a representative set of Pareto optimal solutions, quantify the trade-offs in satisfying the different objectives, and/or find a single solution that satisfies the subjective preferences of a designer. There are numerous ways to categorize the solution methods, such as scalarization versus Pareto methods, a priori versus a posteriori articulation of preferences, and so on. Although there have been many solution techniques developed in the past decades, none of these methods is perfect, and selecting among them depends on the requirements of a particular design situation.

In this chapter, we introduce the basic concepts, the nature of multiobjective problems, and solution techniques for readers who have not been exposed to these topics. We do not intend to provide a comprehensive review nor offer presentations for research-style discussions. We include only sufficient mathematical details to explain concepts and solution techniques, and then use simple examples for illustration. We offer MATLAB scripts for most examples so that readers are able to learn to solve their own multiobjective problems by creating similar scripts. We use similar examples throughout the

chapter while discussing different solution techniques. By doing so, we hope readers are able to gain a better understanding of these methods and see the pros and cons between them. We also revisit the decision theories discussed in Chapter 16 and discuss the application of them to engineering design in the context of multiobjective optimization. Also provided is a short review of the available software tools for use. For a more thorough and comprehensive discussion of multiobjective optimization, readers are referred to excellent books and articles (e.g., Deb, 2001; Miettinen, 1999; Marler and Arora, 2004; Branke et al., 2008).

In addition, we include advanced topics that are relevant to design optimization. We present reliability-based design optimization that incorporates uncertainty of physical parameters and design variables into design optimization, leading to designs with less probability of failure. Also, we present a case of design optimization that takes product cost including manufacturing as an objective function. This case represents a more realistic design scenario that eventually leads to less expensive designs. These topics are open and under active research.

Although we do not get into a discussion of multidisciplinary optimization, we include a single-piston engine example as a tutorial project to illustrate a design scenario that involves multiple engineering disciplines. We use the suite of computer-aided design (CAD), engineering (CAE), and manufacturing (CAM) software in Pro/ENGINEER and SolidWorks and we employ the interactive process discussed in Chapter 17 to proceed with the design.

The objectives of this chapter include (1) introducing readers to the basic concepts and solution techniques of multiobjective optimization, (2) pointing out the pros and cons of different solution techniques and their applicability to practical engineering problems, and (3) offering sample MATLAB scripts as references so that readers will be able to write their own to solve similar problems. We also provide a short discussion on software tools, both academic and commercial, in hope of offering readers basic ideas of selecting proper software tools that are suitable to specific needs. Certainly, the advanced topics discussed at the end of the chapter aim to provide readers with the flavor of those under active research.

19.1 INTRODUCTION

So far, we have considered problems with only one objective function, the so-called single-objective optimization. In practice, many engineering design problems involve more than one objective function. In many situations, these objective functions may be in conflict with one another. For example, it is desirable for a design team to minimize weight while maximizing the strength of a particular structural component, or maximize performance of a vehicle (such as increasing engine torque output for accelerating the vehicle in a shorter time) while minimizing fuel consumption and emission of pollutants. Both are multiobjective optimization (MOO) problems involving two or more objectives. Mathematically, a MOO problem can be formulated as follows:

$$\text{Minimize: } \mathbf{f}(\mathbf{x}) \tag{19.1a}$$

$$\text{Subject to: } g_i(\mathbf{x}) \leq 0, \quad i = 1, m \tag{19.1b}$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, p \tag{19.1c}$$

$$x_k^{\ell} \leq x_k \leq x_k^{u}, \quad k = 1, n \tag{19.1d}$$

This is identical to that of Eq. 17.3 discussed in Chapter 17, except that in Eq. 19.1a, $\mathbf{f}(\mathbf{x})$ is the vector of objective functions, $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_q(\mathbf{x})]^{\mathrm{T}}$, and q is the number of the objective functions.

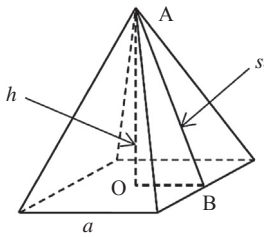


FIGURE 19.1

The pyramid example.

Before entering formal discussion, we present a simple pyramid example to illustrate the basic concept of formulating a multiobjective optimization problem. As shown in Figure 19.1, the base width and height of the pyramid are a and h, respectively. The chord length of the triangle OAB is s. The objective functions of the pyramid design problem are minimizing the lateral surface area A and minimizing the total surface area T by varying two design variables, base width a and height h. The volume of the pyramid must be greater than 1500 in.^{3}, and the upper bounds of the base width a and height h are 30 in. Mathematically, the optimization problem is defined as follows:

$$\begin{aligned} \text{Minimize: } A(a,h) &= 2as = 2a\sqrt{\left(\frac{a}{2}\right)^2 + h^2} \\ T(a,h) &= A + a^2 = 2a\sqrt{\left(\frac{a}{2}\right)^2 + h^2} + a^2 \end{aligned} \tag{19.2a}$$

$$\text{Subject to: } V(a,h) = \frac{a^2 h}{3} \geq 1500 \tag{19.2b}$$

$$0 < a \leq 30 \tag{19.2c}$$

$$0 < h \leq 30 \tag{19.2d}$$

How do we solve this problem? Is there one single solution that minimizes both $A(a, h)$ and $T(a, h)$ and satisfies all the constraints? How do we solve a multiobjective optimization problem, such as the pyramid example defined in Eq. 19.2?

To illustrate the nature of the problem involved in MOO, we present the following simpler example (Example 19.1), in which the problem and solutions can be visualized graphically. We return to the pyramid example in Section 19.2.3 after introducing a few basic solution concepts of MOO.

EXAMPLE 19.1

Solve the following MOO problem, defined as

$$\text{Minimize: } f_1(x_1, x_2) = x_1^2 + x_2^2 \tag{19.3a}$$

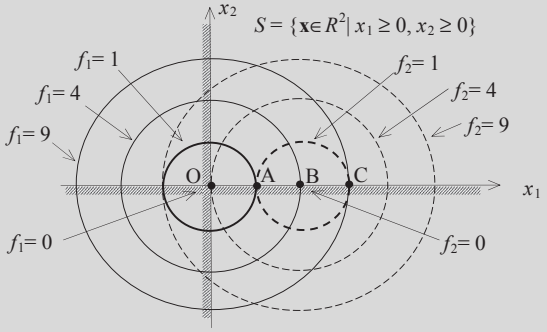
$$f_2(x_1, x_2) = (x_1 - 2)^2 + x_2^2 \tag{19.3b}$$

$$\text{Subject to: } x_1 \geq 0, x_2 \geq 0 \tag{19.3c}$$

EXAMPLE 19.1—cont'd

Solution

The two objective functions f_1 and f_2 can be graphed in the design space, in this case the x_1-x_2 plane, as shown below.



The iso-lines of the objective functions (or objective function contours) f_1 and f_2 are circles with center points (0, 0) and (2, 0), respectively. They are graphed above in solid and dotted lines, respectively. The feasible set (or feasible region) of this problem is defined as:

$$S = \{\mathbf{x} \in R^2 | x_1 \geq 0, x_2 \geq 0\}$$

which is the first quadrant of the x_1-x_2 plane shown in the figure above. It is apparent that the minimum of f_1 is at point O = (0, 0), and the minimum of f_2 is at point B = (2, 0). However, at point O, $f_2 = 4$, and at point B, $f_1 = 4$, which are not their respective minimum. It is obvious that there is no single point that offers both f_1 and f_2 at their minimum. A best possible compromised solution could be at point A = (1, 0), at which $f_1 = f_2 = 1$, which is a compromised solution. Moving away from this point may reduce one objective but result in increasing the other objective. As a matter of fact, any solutions between points O and B may be considered acceptable because reducing one objective function cannot be accomplished without increasing the other objective function. A point that is outside the line segment between points O and B (that is, $x_1 \notin (0, 2)$ and $x_2 = 0$) can be changed to reduce both objective functions simultaneously. For example, at point C = (3, 0), we have $f_1 = 9$ and $f_2 = 1$. By moving from point C to B, both objectives f_1 and f_2 can be reduced. Note that points between O and B, defined as $S_p = \{\mathbf{x} \in R^2 | 0 \leq x_1 \leq 2, x_2 = 0\}$, are called the Pareto optimal set of the MOO problem. A Pareto optimal set usually consists of an infinite number of solutions that are considered legitimate. More about the Pareto optimum is discussed in Section 19.2.

Note that for a simple problem such as Example 19.1, the Pareto optimum can be found easily in the design space. Unfortunately, this is not the case even for a simple problem like the pyramid example. In general, it is insufficient to discuss the Pareto optimum in the design space. To understand the concept of MOO problems and solution techniques, we will have to convert the MOO problem to its criterion (or objective) space. We discuss the criterion space together with concepts and basic terminologies in Section 19.2. We then discuss solution techniques to MOO problems in Section 19.3. Recall that we discussed design examples of multiple objectives in Chapter 2 using both utility and game theories. We will revisit these examples in Section 19.4. In Section 19.5, we present a brief overview of the software tools for solving MOO problems, both academic and commercial. We then present two advanced topics relevant to design optimization in Section 19.6. They are reliability-based design optimization and design optimization for structural performance and manufacturing cost.

19.2 BASIC CONCEPT

The scalar concept of "optimality" of single-objective optimization problems does not apply directly in the multiobjective setting. To understand the concept and solution techniques, the notion of Pareto optimality has to be introduced. In this section, we start by introducing the criterion space by revisiting the simple MOO example of Example 19.1. We then introduce basic terminologies to facilitate our later discussion. With the basic concept discussed, we revisit the pyramid example to reinforce the concept.

19.2.1 CRITERION SPACE AND DESIGN SPACE

In Example 19.1, we depicted the design space of the MOO problem. We defined its feasible set S, plotted the objective function contours, and identified its Pareto optimal set S_p in design space. Alternatively, a MOO problem can be depicted in the criterion space with axes represented by objective functions. For example, the MOO problem of Example 19.1 can be depicted in its criterion space f_1-f_2, as shown in Figure 19.2(b). Note that the side constraints $x_1 \geq 0$ and $x_2 \geq 0$ are translated into the criterion space in this simple example by solving them in terms of f_1 and f_2:

$$\begin{aligned} x_1(f_1,f_2) &= 0.25 \times (f_1 - f_2) + 1 \geq 0 \\ x_2(f_1,f_2) &= \sqrt{f_1 - x_1^2} = \sqrt{f_1 - (0.25 \times (f_1 - f_2) + 1)^2} \geq 0 \end{aligned} \tag{19.4}$$

The feasible criterion space Z is defined simply as the set of objective function values corresponding to the feasible points in the design space:

$$Z = \{f(\mathbf{x}) | \mathbf{x} \in S\} \tag{19.5}$$

As shown in Figure 19.2(a), the points O, A, and B in the design space are mapped into the criterion space at o, a, and b, respectively. The Pareto optimum in the design space $S_p = \{\mathbf{x} \in R^2 |\ 0 \leq x_1 \leq 2, x_2 = 0\}$ is converted into the curve segment oab in the criterion space, defined as $Z_p = \{\mathbf{f} = (f_1, f_2) \in R^2 |\ 0 \leq f_1 \leq 4, 0 \leq f_2 \leq 4, \sqrt{f_1 - (0.25 \times (f_1 - f_2) + 1)^2} = 0\}$. The curve oab in the criterion space is

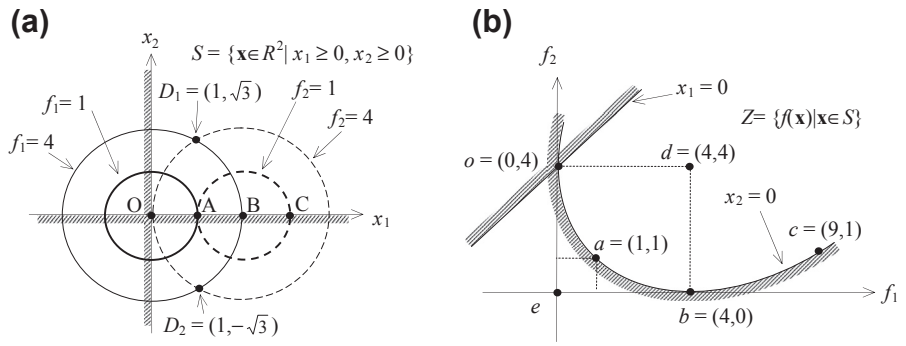


FIGURE 19.2

The example MOO problem depicted in (a) design space and (b) criterion space.

called the Pareto solution, Pareto optimum, Pareto front, or Pareto set, representing the solutions of the MOO problem. It is clearly shown in the criterion space that the minimum of the objective function f_1 is 0 and is located at point $o = (0, 4)$. On the other hand, the minimum of the objective function f_2 is 0 and is located at point $b = (4, 0)$. Moreover, an objective function (f_1 or f_2) cannot be further reduced without increasing the other objective function for any point on the Pareto front. For a point not on the Pareto front, such as point $d = (4, 4)$, it is possible to reduce both objective function values simultaneously by moving the point toward the Pareto front.

A concept related to the feasibility of design points in the design space is that of attainability in criterion space. As discussed in Chapter 17, the feasibility of a design implies that no constraint is violated in the design space. Attainability implies that a point in the criterion space can be related to a point in the design space. It is important to note that each point in the design space can be mapped to a point in the criterion space. However, the reverse may not be true; that is, every point in the criterion space does not necessarily correspond to points or a single point in the design space. For example, point $e = (0, 0)$ in the criterion space does not map back to any point in the design space. Also, point $d = (4, 4)$ in the criterion space maps two points $D_1 = (1, \sqrt{3})$ and $D_2 = (1, -\sqrt{3})$ in the design space, and $D_1 \in S$ and $D_2 \notin S$. We are only interested in the points in the criterion space that are attainable.

19.2.2 PARETO OPTIMALITY

The concept in defining solutions for MOO problems is that of Pareto optimality. A point $\mathbf{x}^*$ in the feasible design space S is called Pareto optimal if there is no other point in the set S that reduces at least one objective function without increasing another one. In Example 19.1, the Pareto optimal is $\mathbf{x}^* \in S_p = \{\mathbf{x} \in R^2 | \ 0 \le x_1 \le 2, x_2 = 0\}$. Pareto optimal is defined more precisely as follows.

Definition 1: Pareto Optimal. A point $\mathbf{x}^* \in S$ is Pareto optimal iff (i.e., if and only if) $\nexists$ (there does not exist) another point $\mathbf{x} \in S$ such that $f_i(\mathbf{x}) \le f_i(\mathbf{x}^*)$ $\forall$ (for all) i and $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$ for least one i.

To illustrate the above statement, we assume that $\mathbf{x}^* = (3, 0)$ is a Pareto optimal (in fact, it is point C in Figure 19.2(a) and we know point C is not Pareto optimal). At this point, $f_1(\mathbf{x}^*) = 9$ and $f_2(\mathbf{x}^*) = 1$. Let us see if $\mathbf{f}(\mathbf{x}) \le \mathbf{f}(\mathbf{x}^*)$ and $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$ for at least one i are true. We can easily find many points to test the conditions. For example, we pick point A, where $\mathbf{x} = (1, 0)$, and $f_1(\mathbf{x}) = f_2(\mathbf{x}) = 1$. Hence, $\mathbf{f}(\mathbf{x}) \le \mathbf{f}(\mathbf{x}^*)$ with $f_1(\mathbf{x}) = 1 < f_1(\mathbf{x}^*) = 9$. Therefore, from Definition 1, $\mathbf{x}^* = (3, 0)$ is not Pareto optimal. On the other hand, if we pick $\mathbf{x}^* = (1, 0)$ (point A), $f_1(\mathbf{x}^*) = 1$ and $f_2(\mathbf{x}^*) = 1$. There does not exist any other point where f_1 and f_2 are less than or equal to those of (1, 0). Hence, $\mathbf{x}^* = (1, 0)$ is a Pareto optimal. In fact, all points in S_p satisfy Definition 1; hence, all are Pareto optimal.

It is important to note that the Pareto optimal set Z_p is always on the boundary of the feasible criterion space Z. When there are just two objective functions, as shown in Example 19.1, the minimum points of individual objective functions define the endpoints of the Pareto front (i.e., points o and b in Figure 19.2(b)), assuming the minima to be unique.

Although the Pareto optimal set is always on the boundary of Z, it is not necessarily defined by the constraints. If the MOO problem of Example 19.1 is redefined as a nonconstrained problem by removing the side constraints $x_1 \ge 0, x_2 \ge 0$, how do we find the Pareto optimal? In this case, the Pareto optimal is defined by the relationship between the gradients of the objective functions. For cases of two objective functions, the gradients of the objective functions point in opposite directions at all Pareto optimal points (Arora, 2012). For the problem in Example 19.1 without side constraints, the Pareto optimal is the line connecting the two centers of the objective function contours, which is unchanged

from the constrained problem. From Figure 19.2(b), the Pareto optimal can be easily identified on the curve segment oab.

A concept closely related to Pareto optimality is that of weak Pareto optimality. At the weak Pareto optimal points, it is possible to improve some objective functions without penalizing others. A weak Pareto optimal point is defined as follows.

Definition 2: Weak Pareto Optimal. A point $\mathbf{x}^* \in S$ is weakly Pareto optimal iff $\nexists$ another point $\mathbf{x} \in S$ such that $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)\ \forall\ i$.

In another words, a point is weakly Pareto optimal if there is no other point that improves all objective functions simultaneously. However, there may be points that improve some of the objectives while keeping others unchanged.

The concept of weak Pareto optimality is illustrated in Figure 19.3. We minimize two objectives f_1 and f_2. Note that lines AB and BC are the boundary of the feasible criterion space Z and are respectively horizontal and vertical. In this case, all points on line A–B–C are weakly Pareto optimal. However, only points A and C are Pareto optimal.

If we take any point on AB (not including A), such as point E, we cannot find another point $\mathbf{x}$ in the feasible criterion space such that $f_1(\mathbf{x}) < f_1(\mathbf{x}_E)$ and $f_2(\mathbf{x}) < f_2(\mathbf{x}_E)$; therefore, point E is weakly Pareto optimal. On the other hand, we can find at least one point $\mathbf{x}$ (such as all points between AE) such that $f_1(\mathbf{x}) < f_1(\mathbf{x}_E)$ and $f_2(\mathbf{x}) = f_2(\mathbf{x}_E)$; therefore, point E is not Pareto optimal. Similarly, all points on BC (not including C) are weakly Pareto optimal but not Pareto optimal. Pareto optimal is weakly Pareto optimal, but a weakly Pareto optimal is not necessarily Pareto optimal.

Another common concept is that of nondominated and dominated points, which is defined as follows.

Definition 3: Nondominated and Dominated. A vector of objective functions $\mathbf{f}^* = \mathbf{f}(\mathbf{x}^*) \in Z$ is nondominated iff $\nexists$ another vector $\mathbf{f} \in Z$ such that $f_i \leq f_i^*\ \forall\ i$ and $f_i < f_i^*$ for least one i. Otherwise, $\mathbf{f}^*$ is dominated.

Pareto optimality generally refers to both the design and the criterion spaces. In numerical algorithms, the idea of nondomination in the criterion space is often used for a subset of points; one point may be nondominated compared with other points in the subset. A Pareto optimal point has no other point that improves at least one objective without detriment to another; that is, it is nondominated.

A unique point, called utopia point or ideal point, in the criterion space is defined as follows.

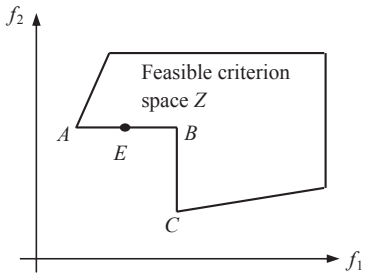


FIGURE 19.3

Illustration of weak Pareto optimality.

Definition 4: Utopia Point. A point $\mathbf{f}^{\mathrm{o}}$ in the criterion space is called the utopia point if $f_i^o = \min\{f_i(\mathbf{x}) \,|\mathbf{x} \in S\}$, $i = 1$ to q, and q is the number of the objective functions.

The utopia point is obtained by minimizing each objective function without regard for other objective functions. Each minimization yields a design point in the design space and the corresponding value for the objective function. As shown in Figure 19.2(b), point e is the utopia point of Example 19.1.

In general, it is rare that each minimization will end up at the same point in the design space. That is, one design point cannot simultaneously minimize all of the objective functions. Thus, the utopia point exists only in the criterion space and, in general, it is not attainable, such as the utopia point e in Figure 19.2(b).

The next best thing to a utopia point is a Pareto solution that is as close as possible to the utopia point. Such a solution is called a compromise solution, for example, point a in Figure 19.2(b). The term closeness can be defined in several different ways. Usually, it implies that one minimizes the Euclidean distance $D(\mathbf{x})$ from the utopia point in the criterion space, which is defined as follows:

$$D(\mathbf{x}) = ||\mathbf{f}(\mathbf{x}) - \mathbf{f}^{\mathrm{o}}|| = \sqrt{\sum_{i=1}^{q} \left[f_i(\mathbf{x}) - f_i^o\right]^2} \tag{19.6}$$

where f_i^o represents a component of the utopia point in the criterion space. Compromise solutions are Pareto optimal.

19.2.3 GENERATION OF PARETO OPTIMAL SET

For a simple problem such as Example 19.1, we can easily solve it by translating the problem from design space to criterion space and identifying its Pareto optimal set in the criterion space as $Z_p = \{\mathbf{f} = (f_1, f_2) \in R^2 |\, 0 \le f_1 \le 4, 0 \le f_2 \le 4, \sqrt{f_1 - (0.25 \times (f_1 - f_2) + 1)^2} = 0\}$. The Pareto optimal set can also be translated back to its design space $S_p = \{\mathbf{x} \in R^2 |\, 0 \le x_1 \le 2, x_2 = 0\}$. For problems just a bit more complicated, such as the pyramid design, none of the above is possible. So, how do we solve MOO problems in general?

We first use a brute-force approach—the generative method similar to that discussed in Chapter 17—to solve for the pyramid problem. We will then make a few comments before moving into the next section for the discussion of general solution techniques.

Recall the problem definition of the pyramid problem in Eq. 19.2. The feasible set of the problem is defined in its design space (in this case, the a-h plane) as

$$S = \left\{(a,h) \in R^2 | V(a,h) = \frac{a^2 h}{3} \ge 1,500; \quad \text{and} \quad 0 < a \le 30, 0 < h \le 30\right\} \tag{19.7}$$

The feasible set S is graphed in Figure 19.4, with optimal points $\mathbf{x}_t^* = (14.7, 20.8)$ and $\mathbf{x}_a^* = (18.5, 13.1)$ for objective functions T and A, respectively. Note that the optimal points $\mathbf{x}_t^*$ and $\mathbf{x}_a^*$ are obtained by solving the two respective single-objective problems using MATLAB (Script 19.1 on the book's companion website, http://booksite.elsevier.com/9780123820389).

Translating this simple problem from design space to its criterion space is not straightforward. In fact, it is very difficult, if not entirely impossible, to write the design variables a and h as functions

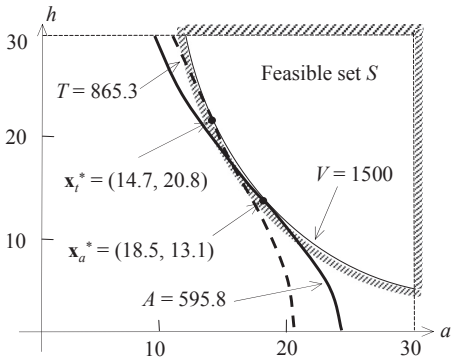


FIGURE 19.4

The design space of the pyramid example with feasible set and optimal points $\mathbf{x}_t^*$ and $\mathbf{x}_a^*$.

of objective functions A and T analytically. Therefore, we use the generative method to graph feasible and infeasible solutions in criterion space.

We first generate 1,000,000 random points for a and h, respectively. We then calculate the numerical values of the two objective functions A and T for each point (a, h). In the meantime, we calculate volume V. If V is greater or equal to 1500, the design represented in the point (a, h) is feasible; otherwise, it is infeasible. We store feasible and infeasible points separately and graph them on the criterion space in different colors. The points at the junction of the areas of different colors are the boundary of the feasible criterion space Z. Once the feasible criterion space is identified, we locate the minimum points of functions A and T and identify the Pareto front. The MATLAB script that graphs the criterion space and Pareto front shown in Figure 19.5 can be found on the book's companion website, http://booksite.elsevier.com/9780123820389 (Script 19.2). The MATLAB script also found that the minima of A and T are, respectively, 594.8 and 865.5. The minimum A occurs at $a = 18.63$ and $h = 12.97$, which is very close to the $\mathbf{x}_a^*$ found earlier (see Figure 19.4). The minimum T occurs at $a = 14.56$ and $h = 21.23$, which is again very close to $\mathbf{x}_t^*$.

In fact, we ran six cases with sample points ranging from 100 to 10,000,000, as shown in Table 19.1. The results show that for sample points more than 10,000, the results do not vary significantly. However, if we graph the criterion space with a sample size of 10,000, the boundary of the feasible criterion set, hence the Pareto front, cannot be clearly identified (see Figure 19.6).

Although the generative method is simple, it works well only for simple problems, such as the pyramid example. In general, engineering design problems involve significant computation efforts for function evaluations, and it is impractical to find Pareto front using the generative method in general. On the other hand, do we really need to solve for an entire set of the Pareto front in order to make adequate design decisions? Instead of the generative method, are there plausible methods that are practical for support of engineering design involving multiple objectives?

We discuss solution techniques for multiobjective optimization problems in the next section. Note that a key characteristic of MOO solution techniques is the nature of the solutions that they provide. Some methods always yield Pareto optimal solutions but may skip certain points in the Pareto optimal set; that is, they may not be able to yield all or most of the Pareto optimal points. Other methods are

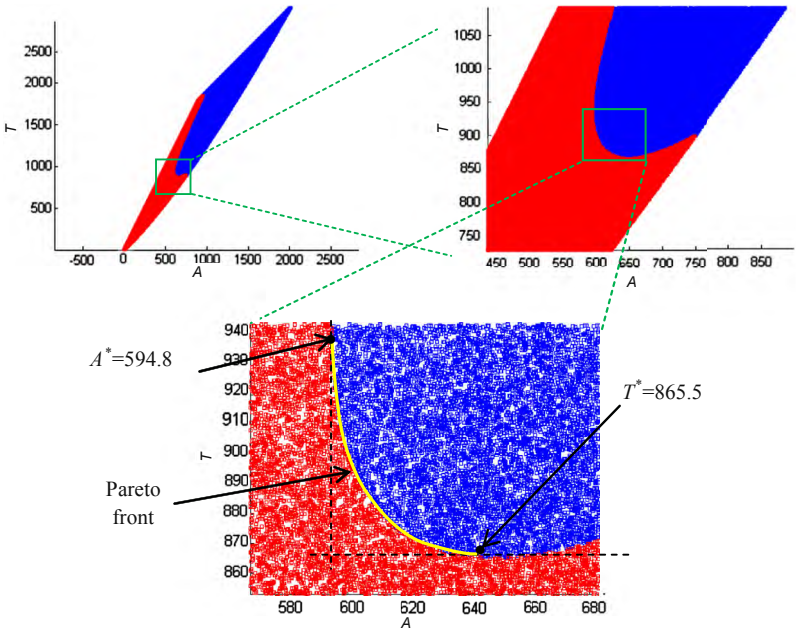


FIGURE 19.5

The criterion space of the pyramid example shown for 1,000,000 sample points.

Table 19.1 Cases of the Pyramid Example with Different Sample Sizes Ranging from 100 to 10,000,000

	Surface Area A			Total Area T		
Sample Points	**Minimum A**	***a***	***h***	**Minimum T**	***a***	***h***
10,000,000	594.82	18.486	13.168	865.35	14.675	20.895
1,000,000	594.85	18.627	12.970	865.54	14.559	21.232
100,000	595.68	18.924	12.577	866.26	14.999	20.021
10,000	595.57	18.368	13.360	866.23	14.724	20.787
1000	612.16	19.199	12.728	876.89	15.021	20.336
100	643.25	20.499	11.878	901.44	15.416	20.100

able to capture most points in the Pareto optimal set, but they may also provide non-Pareto optimal points. In any case, the primary goal of solving a multiobjective optimization problem is to support the designer to make adequate design decisions, which may involve the designer's preferences in ordering or setting the relative importance of individual objective functions. A designer may articulate these preferences before solving the MOO problem or wait until sufficient solutions become available and then set the preferences. In some cases, a designer may not have any preferences at all. Different solution techniques are suitable for the respective situations.

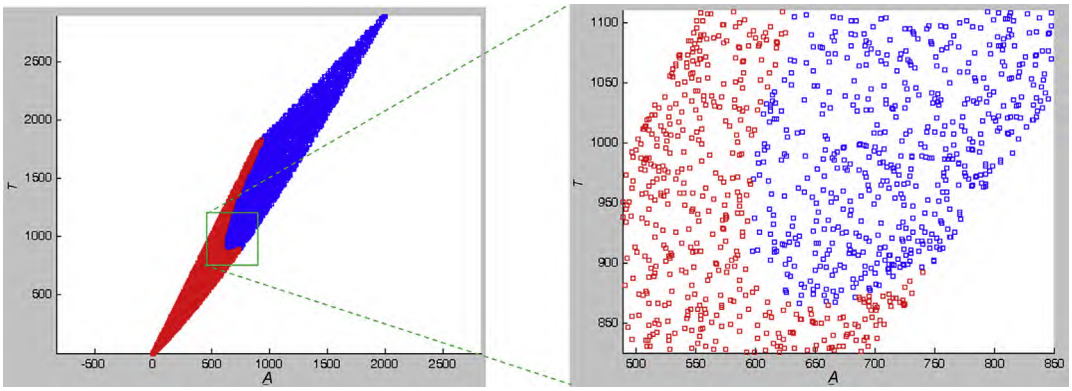


FIGURE 19.6

The criterion space of 10,000 sample points.

19.3 SOLUTION TECHNIQUES

As discussed earlier, the brute-force approach (e.g., the generative method) requires too many function evaluations; as a result, it is impractical for solving general engineering problems. Numerous solution techniques have been proposed, with consideration of minimizing the number of function evaluations, among other factors. Because a primary goal of multiobjective optimization is to model a designer's preferences (ordering or relative importance of objectives and goals), methods are categorized depending on how the designer articulates these preferences: methods with a priori articulation of preferences, methods with a posteriori articulation of preferences, and methods with no articulation of preferences. Section 19.3.2 contains methods that involve a priori articulation of preferences, which implies that the designer indicates the relative importance of the objective functions or desired goals before running the optimization algorithm. With the preferences specified, the MOO is converted to a single optimization problem, leading to a single solution. Section 19.3.3 describes methods with a posteriori articulation of preferences, which offer a set of solutions that allow the designer to choose. In Section 19.3.4, methods that require no articulation of preferences are addressed. Although methods based on genetic algorithms (GA) generate multiple solutions for designers to choose, the concept and solution techniques are somehow different than those of conventional methods with a posteriori articulation of preferences. Therefore, we discuss GA-based methods separately in Section 19.3.5.

19.3.1 NORMALIZATION OF OBJECTIVE FUNCTIONS

Many multiobjective optimization methods involve comparing and making decisions about different objective functions. However, values of different functions may have different units and/or significantly different orders of magnitude, making comparisons difficult. Thus, it is usually necessary to transform the objective functions such that they all have similar orders of magnitude. Although there are different approaches proposed for such a purpose, one of the simplest approaches is to normalize individual objective functions by their respective absolute function values at current (or initial) designs:

$$f_i^{\text{norm}}(\mathbf{x}) = \frac{f_i(\mathbf{x})}{|f_i(\mathbf{x}^0)|} \tag{19.8}$$

where f_i^{norm} is the ith normalized objective function, and $\mathbf{x}^0$ is the vector of design variables at current or initial design. This method ensures that all objective functions are normalized to 1 or -1 to start with. Certainly, we assume that $f_i(\mathbf{x}^0)$ is not zero or close to zero at the initial design and throughout the optimization process. Note, however, that if all of the objective functions have similar values, such as the pyramid example, normalization may not be needed.

In the following sections, we assume that the objective functions have been normalized in a certain way if necessary.

19.3.2 METHODS WITH A PRIORI ARTICULATION OF PREFERENCES

The methods to be discussed in this section allow the designer to specify preferences, which may be articulated in terms of goals or the relative importance of different objectives. Most of these methods incorporate parameters, which are coefficients, exponents, constraint limits, and so on, that can either be set to reflect designer preferences or be continuously altered in an attempt to find multiple solutions that roughly represent the Pareto optimal set.

19.3.2.1 Weighted-Sum Method

A multiobjective optimization problem is often solved by combining its multiple objectives into one single-objective scalar function. The simplest and most common approach is the weighted-sum or scalarization method, defined as

$$\underset{\mathbf{x}\in S}{\text{Minimize}} \quad u(\mathbf{x}) = \sum_{i=1}^{q} w_i f_i(\mathbf{x}) \tag{19.9}$$

which represents a new optimization problem with a unique objective function $u(\mathbf{x})$. Note that the weights w_i's are typically set by the decision maker, such that $\sum_{i=1}^{q} w_i = 1$ and $w_i \geq 0 \ \forall \ i$. Graphically, the new objective function $u(\mathbf{x})$ is a hyperplane in the criterion space of q dimensions. For a two-objective problem ($q = 2$), the new objective function $u(\mathbf{x})$ is a straight line in the f_1-f_2 plane. Note that the slope of the straight line is determined by weights w_1 and w_2. The optimal solution is the tangent point of the straight line intersecting with the Pareto front of the feasible criterion space, as shown in Figure 19.7.

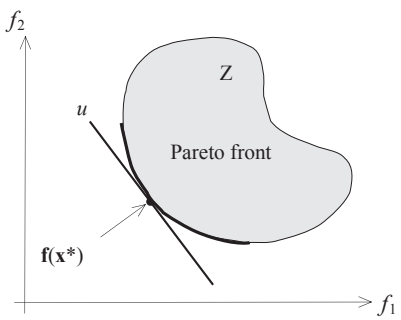


FIGURE 19.7

Geometrical representation of the weighted-sum approach in the case of a convex Pareto front.

We introduce a simple example in Example 19.2 to illustrate the method. This example is formulated as a multiobjective linear programming (MOLP) problem that simplifies the mathematical calculations. We use two design variables to facilitate graphical presentation.

EXAMPLE 19.2

Solve the following MOLP problem using the weighted-sum method,

$$\text{Minimize: } f_1 = 2x_1 - 3x_2 \tag{19.10a}$$

$$f_2 = 2x_1 + x_2 \tag{19.10b}$$

$$\text{Subject to: } g_1 = 7 - x_1 - 5x_2 \le 0 \tag{19.10c}$$

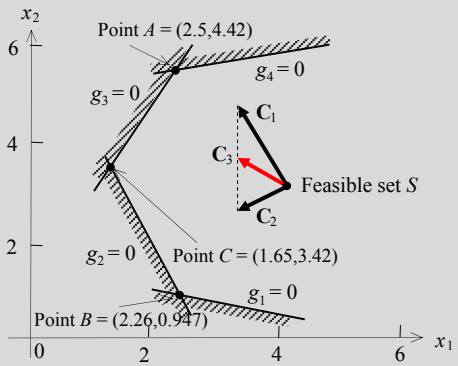
$$g_2 = 10 - 4x_1 - x_2 \le 0 \tag{19.10d}$$

$$g_3 = -7x_1 + 6x_2 - 9 \le 0 \tag{19.10e}$$

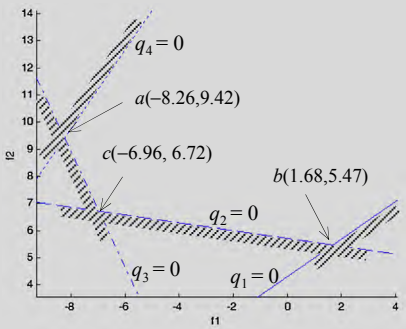
$$g_4 = -x_1 + 6x_2 - 24 \le 0 \tag{19.10f}$$

Solution

We define the vectors of steepest descent direction (negative gradients) of the objective functions f_1 and f_2 in the design space as $\mathbf{C}_1 = (-2, 3)$ and $\mathbf{C}_2 = (-2, -1)$, respectively. Note that $\mathbf{C}_1$ is a vector that defines the direction on the x_1-x_2 plane that minimizes f_1, as is $\mathbf{C}_2$ for function f_2. The feasible set S, as well as the vectors $\mathbf{C}_1$ and $\mathbf{C}_2$, are shown in the left figure below.



Design space



Criterion space

It is obvious that the minimum of $f_1 = -8.26$ (as if $w_1 = 1$, $w_2 = 0$) is found at point $A = (2.5, 4.42)$ (the intersecting point of $g_3 = 0$ and $g_4 = 0$; and $f_2 = 9.42$ at this point), and minimum of $f_2 = 5.47$ (as if $w_1 = 0$, $w_2 = 1$) is at point $B = (2.26, 0.947)$ (the intersecting point of $g_1 = 0$ and $g_2 = 0$; and $f_1 = 1.68$ at this point).

For any $0 < w_i < 1$ and $w_1 + w_2 = 1$, the gradient $\mathbf{C}_3$ of the weighted-sum function $u(\mathbf{x})$ points in a direction that is between $\mathbf{C}_1$ and $\mathbf{C}_2$ (e.g., $w_1 = w_2 = 0.5$, as shown in the left figure above). Therefore, the minimum of the weighted-sum function is at point $C = (1.65, 3.42)$ (the intersecting point of $g_2 = 0$ and $g_3 = 0$), at which $f_1 = -6.96$, $f_2 = 6.72$.

The MOLP problem can be easily translated into its criterion space as shown in the right figure above. Note that the constraint functions in the criterion space are, respectively,

$$q_1 = 9f_1 - 13f_2 + 56 \le 0 \tag{19.11a}$$

EXAMPLE 19.2—cont'd

$$q_2 = -f_1 - 7f_2 + 40 \leq 0 \tag{19.11b}$$

$$q_3 = -19f_1 - 9f_2 - 72 \leq 0 \tag{19.11c}$$

$$q_4 = -13f_1 + 9f_2 - 192 \leq 0 \tag{19.11d}$$

The feasible criterion space is graphed. The three vertices of the feasible criterion space $a = (-8.26, 9.42)$, $b = (1.68, 5.47)$, and $c = (-6.96, 6.72)$ are the intersection points of $q_3 = 0$ and $q_4 = 0$, $q_1 = 0$ and $q_2 = 0$, as well as $q_2 = 0$ and $q_3 = 0$, respectively. The Pareto front consists of all points on the line segments ac and cb. All three solutions obtained (for $w_1 = 1$, $w_2 = 0$; $w_1 = 0$, $w_2 = 1$; and $w_1 = w_2 = 0.5$) are Pareto optimum.

If we set $w_1 = 19/28$ and $w_2 = 9/28$, then $\mathbf{C}_3 = w_1\mathbf{C}_1 + w_2\mathbf{C}_2 = (-2, 12/7)$, which is the gradient of $g_3 = 0$. In this case, the solutions contain points on the line segment between A and C on $g_3 = 0$ in the design space. The optimum is not unique. In the criterion space, the solutions are line segment ac, which is part of the Pareto front.

The relative value of the weights generally reflects the relative importance of the objectives. The weights can be used in two ways. The designer may either set w_i to reflect preferences before the problem is solved or systematically alter weights to yield different Pareto optimal points. In fact, most methods that involve weights can be used in both of these capacities—to generate a single solution or multiple solutions. We demonstrate this statement in Example 19.3.

EXAMPLE 19.3

Solve the following MOO problem.

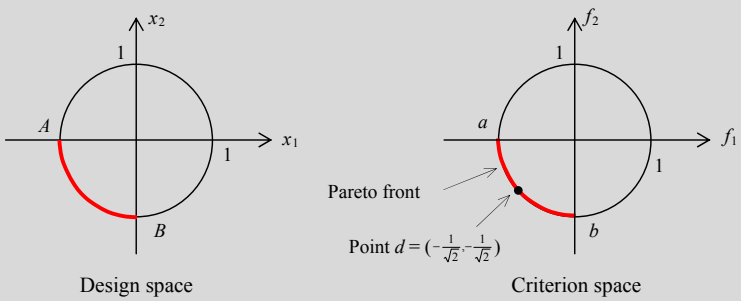
$$\text{Minimize: } f_1 = x_1 \tag{19.12a}$$

$$f_2 = x_2 \tag{19.12b}$$

$$\text{Subject to: } x_1^2 + x_2^2 - 1 = 0 \tag{19.12c}$$

Solution

This problem has identical design space and criterion space, as shown below.



Design space Criterion space

It is obvious that the Pareto front is the circular arc ab (including points a and b) in the criterion space. The negative gradients of the objective functions f_1 and f_2 are $\mathbf{C}_1 = (-1, 0)$ and $\mathbf{C}_2 = (0, -1)$, respectively. If we choose $w_1 = 1/2$ and

Continued

EXAMPLE 19.3—cont'd

$w_2 = 1/2$, then $\mathbf{C}_3 = (-1/2, -1/2)$. If we solve for this single-objective optimization problem, we should obtain a solution at point $d = \left(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right)$. In general, the Pareto optimal set can be obtained by choosing different combinations of weights w_1 and w_2 as many times as desired. For this simple example, when the weights are properly selected—for example, $(w_1, w_2) = (0.1, 0.9), (0.2, 0.8), \ldots, (0.9, 0.1)$—the Pareto optimal points are evenly distributed, which is desirable. This is because the Pareto front in this case is a 90° circular arc, whose curvature is constant. Note that, in general, evenly distributed Pareto points may not be possible using the weighted-sum method when the curvature of the front is varying.

Examples 19.2 and 19.3 are solved graphically with minimum calculations. We revisit the pyramid problem discussed in Sections 19.1 and 19.2 to illustrate a more realistic problem-solving scenario in Example 19.4.

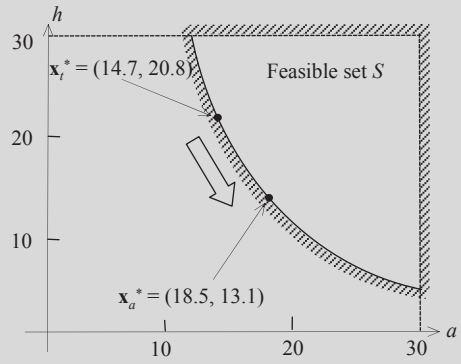
EXAMPLE 19.4

Solve the pyramid problem using the weighted-sum method.

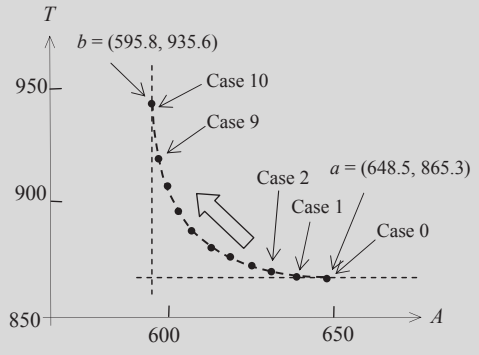
Solution

Although the Pareto front of the pyramid problem has been solved using the generative method and is shown in Figure 19.5, we assume this front is unknown.

We re-sketch below (left) Figure 19.4 of the design space of the pyramid example with feasible set S and optimal points $\mathbf{x}_t^*$ and $\mathbf{x}_a^*$. The two optimal points of the respective single-objective optimization problems are $\mathbf{x}_t^* = (14.7, 20.8)$ and $\mathbf{x}_a^* = (18.5, 13.1)$, respectively. At $\mathbf{x}_t^*$, the two function values are $A(\mathbf{x}_t^*) = A(14.7, 20.8) = 648.5$ and $T(\mathbf{x}_t^*) = T(14.7, 20.8) = 865.3$. Similarly, at $\mathbf{x}_a^*$, the two function values are $A(\mathbf{x}_a^*) = A(18.5, 13.1) = 595.8$ and $T(\mathbf{x}_a^*) = T(18.5, 13.1) = 935.6$. The two optimal points $a = (A(\mathbf{x}_t^*), T(\mathbf{x}_t^*))$ and $b = (A(\mathbf{x}_a^*), T(\mathbf{x}_a^*))$ are pointed out in the criterion space shown below (right) for illustration.



Design space



Criterion space

Using the weighted-sum method, the MOO problem is converted into a single-objective optimization as

$$\underset{(a,h)\in S}{\text{Minimize}}\ u(a,h) = w_a A(a,h) + w_t T(a,h) = w_a\left[2a\sqrt{\left(\frac{a}{2}\right)^2 + h^2}\ \right] + w_t\left[2a\sqrt{\left(\frac{a}{2}\right)^2 + h^2} + a^2\right] \tag{19.13a}$$

EXAMPLE 19.4—cont'd

where the feasible space S is defined in Eq. 19.7. The single-objective problem can be solved, for example, using MATLAB function `fmincon` (Script 19.3). We choose $w_a = 0.0, 0.1, 0.2, \ldots, 1.0$ (and $w_t = 1 - w_s$) and solve for the 11 cases. The results are listed in the following table.

	Weights		Design Point		Surface Area	Total Area	Weighted Objective
Case No.	w_a	w_t	a	h	A	T	u
0	0.0	1.0	14.71	20.80	649.0	865.3	865.3
1	0.1	0.9	14.99	20.02	641.0	865.8	843.3
2	0.2	0.8	15.29	19.24	633.3	867.1	820.4
3	0.3	0.7	15.61	18.47	626.0	869.6	796.7
4	0.4	0.6	15.95	17.69	619.0	873.4	771.6
5	0.5	0.5	16.31	16.92	612.6	878.6	745.6
6	0.6	0.4	16.69	16.15	606.9	885.6	718.4
7	0.7	0.3	17.10	15.38	602.0	894.6	689.8
8	0.8	0.2	17.54	14.62	598.2	906.1	659.8
9	0.9	0.1	18.01	13.86	595.7	920.4	628.2
10	1.0	0.0	18.53	13.10	594.8	938.2	594.8

From the solutions of the 11 cases, it is clear that in the design space, as w_a increases from 0 to 1, the design point moves from $\mathbf{x}_t^*$ to $\mathbf{x}_a^*$. In the criterion space, the function values of A and T are plotted. A dotted curve that connects these points approximates Pareto front accurately. Also, as w_a increases from 0 to 1, the objective point moves from points a to b, indicating the influence of the weight w_a to the solutions of the problem using the weighted-sum method. Essentially, increasing w_a results in smaller values of the objective function A, pushing the solutions in the criterion space from points a to b. Although, as shown in the figure (right, on the previous page), the Pareto points seem to be fairly evenly distributed, in general, it may not be the case if the Pareto front is of significantly varying curvature.

As shown in the above examples, the weighted-sum method is easy to use. If all weights are positive, the minimum of Eq. 19.9 is always a Pareto optimal. However, there are a few recognized difficulties with the weighted-sum method (Arora, 2012). First, even with some of the methods discussed in the literature for determining weights, a satisfactory a priori weight selection does not necessarily guarantee that the final solution will be acceptable; one may have to resolve the problem with different weights. In fact, this is true of most weighted methods. The second problem is that it is impossible to obtain points on nonconvex portions of the Pareto optimal set in the criterion space. This is illustrated in Figure 19.8, which shows feasible criterion space and Pareto front of a two-objective

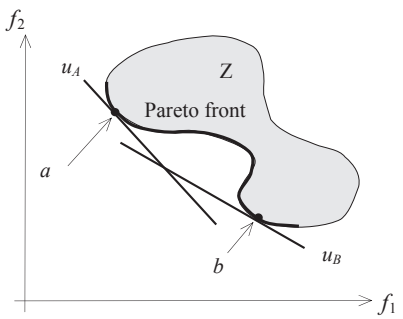


FIGURE 19.8

Geometric illustration of the weighted-sum approach for a nonconvex Pareto front.

problem. In this case, the Pareto front is a concave curve. Using the weighted-sum method, as discussed earlier, the converted objective function $u(\mathbf{x})$ is a straight line in the f_1-f_2 plane. We select two sets of weights to create two new objective functions $u_A(\mathbf{x})$ and $u_B(\mathbf{x})$; both are straight lines with slopes determined by the respective weights, as shown in Figure 19.8. The two straight lines intersect the Pareto front at points a and b, respectively, resulting in two Pareto points. However, it is apparent that a straight line would never be able to reach the concave portion of the Pareto front. Therefore, the weighted-sum method is not able to find any solution located in the concave Pareto front. Another difficulty with the weighted-sum method is that varying the weights consistently and continuously may not necessarily result in an even distribution of Pareto optimal points and an accurate, complete representation of the Pareto optimal set.

An improvement to the weighted-sum is called the weighted-exponential sum, in which exponential p is added to objective functions as

$$\min_{\mathbf{x}\in S} u(\mathbf{x}) = \sum_{i=1}^{q} w_i (f_i(\mathbf{x}))^p \tag{19.14}$$

where $\sum_{i=1}^{q} w_i = 1$ and $w_i \geq 0 \ \forall$ i, and $p > 0$. In this case, p can be thought of as a compensation parameter; that is, a larger p implies that one prefers solutions with both very high and very low objective values rather than those with averaged values. In general, p may need to be very large to capture Pareto points in the nonconvex regions.

19.3.2.2 Weighted Min–Max Method

The idea of the weighted min–max method (also called the weighted Tchebycheff method) is to minimize $u(\mathbf{x})$, which represents the "distance" to an ideal utopia point in criterion space and is given as follows:

$$\underset{\mathbf{x}\in S}{\text{Minimize}}\ u(\mathbf{x}) = \max_i \left\{ w_i \left[f_i(\mathbf{x}) - f_i^o \right] \right\} = \underset{\mathbf{x}\in S}{Min} \left(\max_i \left\{ w_i \left[f_i(\mathbf{x}) - f_i^o \right] \right\} \right) \tag{19.15}$$

A common approach to the treatment of Eq. 19.15 is to introduce an additional unknown parameter λ as follows:

$$\underset{\mathbf{x}\in S}{\text{Minimize:}}\ \lambda \tag{19.16a}$$

$$\text{Subject to: } w_i \left[f_i(\mathbf{x}) - f_i^o \right] - \lambda \leq 0, \quad \forall i \tag{19.16b}$$

The concept of solving the MOO problem defined Eq. 19.16 is illustrated in Figure 19.9 with two objective functions f_1 and f_2. First, the first term of Eq. 19.16b represents the length of a line originating from the utopia point and pointing in a direction, determined by the weights w_1 and w_2, toward the feasible criterion set. By minimizing λ, and hence the length of the line segment, the solution leads to a point on the Pareto front while keeping the design to be feasible.

The key advantage of the weighted min–max method is that it is able to provide almost all the Pareto optimal points, even for a nonconvex Pareto front. It is relatively well suited for generating the representative Pareto optimal front (with variation in the weights). However, this method requires the minimization of individual single-objective optimization problems to determine the utopia point, which can be computationally expensive. We demonstrate the weighted min–max method in Example 19.5.

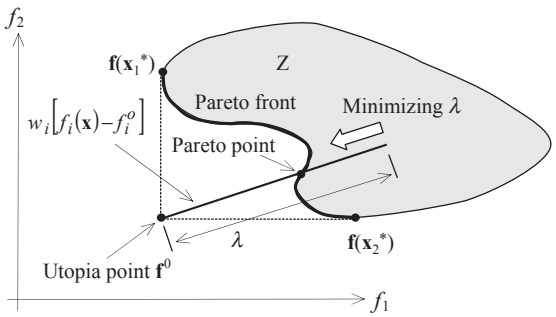


FIGURE 19.9

Geometric illustration of the weighted min–max method.

EXAMPLE 19.5

Solve the same MOO problem of Example 19.3 using the weighted min−max method.

Solution

Following the weighted min−max method, we need to find the utopia point first. In this case, the point is found at $\mathbf{f}^0 = (-1, -1)$, as shown in the figure below, in which we recall that points $a = (-1, 0)$ and $b = (0, -1)$ represent the ends of the Pareto front—in this case, a 90° circular arc.

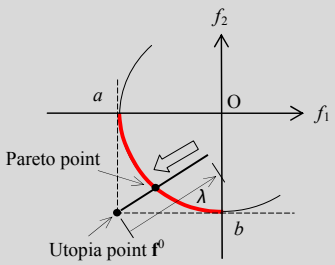
If we choose $w_1 = 1/2$ and $w_2 = 1/2$, then the constraint function of Eq. 19.16b becomes

$$w_i\left[f_i(\mathbf{x}) - f_i^o\right] - \lambda = w_1\left(f_1(\mathbf{x}) - f_1^o\right) + w_2\left(f_2(\mathbf{x}) - f_2^o\right) - \lambda = \frac{1}{2}(x_1 + 1) + \frac{1}{2}(x_2 + 1) - \lambda \leq 0$$

Hence, using the weighted min−max method, we are essentially solving the following single-objective optimization problem:

$$\underset{\mathbf{x}\in S}{\text{Minimize:}}\ \lambda \tag{19.17a}$$

$$\text{Subject to:}\ \frac{1}{2}(x_1 + 1) + \frac{1}{2}(x_2 + 1) - \lambda \leq 0 \tag{19.17b}$$



The single-objective problem can be solved, for example, using MATLAB (Script 19.4). The solutions are obtained as $x_1 = x_2 = -0.7071$, and $\lambda = 0.2929$, which is on the Pareto front. We may adjust the weights to obtain other Pareto points. For example, if we choose $w_1 = 0.1$ and $w_2 = 0.9$, Eq. 19.17b becomes $0.1x_1 + 0.9x_2 + 1 - \lambda \leq 0$, and solutions are $x_1 = -0.1104$, $x_2 = -0.9939$, and $\lambda = 0.0945$, which is another Pareto point.

19.3.2.3 Lexicographic Method

With the lexicographic method, preferences are imposed by ordering the objective functions according to their importance or significance, rather than by assigning weights. After we arrange the objective functions by importance, the most important objective is solved first as a single-objective problem. The second objective is then solved as again a single-objective problem with an added constraint, defined as $f_1(\mathbf{x}) \leq f_1(\mathbf{x}_1{}^*)$, in which $\mathbf{x}_1{}^*$ is the optimal solution of the first objective function. The process is repeated, in which optimal solution obtained in the previous step is added as a new constraint, and the sequence of single-objective optimization problems is solved, one problem at a time. Mathematically, the lexicographic method is defined as

$$\underset{\mathbf{x}\in S}{\text{Minimize: }} f_i(\mathbf{x}) \tag{19.18a}$$

$$\text{Subject to: } f_j(\mathbf{x}) \leq f_j\left(\mathbf{x}_j^*\right); \quad j = 1, i-1; \quad i = 1, q \tag{19.18b}$$

where i represents a function's position in the preferred sequence, and $f_j(\mathbf{x}_j^*)$ represents the minimum value for the jth objective function, found in the jth optimization problem. Note that after the first iteration, $(j > 1)$, $f_j(\mathbf{x}_j^*)$ is not necessarily the same as the independent minimum of $f_j(\mathbf{x})$ because new constraints are introduced for each problem. The algorithm terminates once a unique optimum is determined. Generally, this is indicated when two consecutive optimization problems yield the same solution point. However, determining if a solution is unique (within the feasible design space S) can be difficult, especially with gradient-based solution techniques.

For this reason, often with continuous problems, this approach terminates after simply finding the optimum of the first objective $f_1(\mathbf{x})$. Thus, it is best to use a non-gradient solution technique with this approach. In any case, the solution is, theoretically, always Pareto optimal. We use an example problem (Example 19.6) similar to that of Example 19.2 to illustrate the concept of the method.

EXAMPLE 19.6

Solve the following MOLP problem using the lexicographic method, defined as

$$\text{Minimize: } f_1 = 4x_1 + x_2 \tag{19.19a}$$

$$f_2 = 2x_1 + x_2 \tag{19.19b}$$

$$f_3 = x_1 + x_2 \tag{19.19c}$$

$$\text{Subject to: } g_1 = 7 - x_1 - 5x_2 \leq 0 \tag{19.19d}$$

$$g_2 = 10 - 4x_1 - x_2 \leq 0 \tag{19.19e}$$

$$g_3 = -7x_1 + 6x_2 - 9 \leq 0 \tag{19.19f}$$

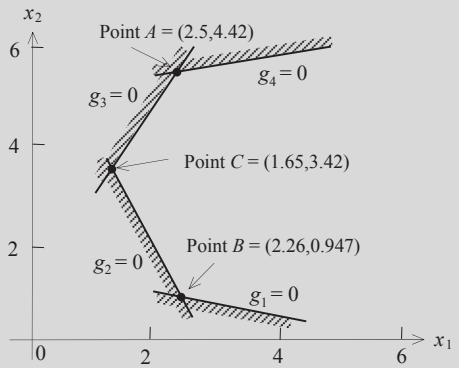
$$g_4 = -x_1 + 6x_2 - 24 \leq 0 \tag{19.19g}$$

EXAMPLE 19.6—cont'd

Solution

Note that the feasible set of the problem defined by Eqs 19.19d–19.19g is identical to that of Example 19.2; that is, $S = \{\mathbf{x} \in R^2 |\ g_i(\mathbf{x}) \leq 0,\ i = 1{,}4\}$. In this problem, we have three objective functions f_1, f_2, and f_3. We arrange the objective functions by importance as f_1, f_2, and f_3. Therefore, we first optimize a single-objective problem for objective f_1 as

$$\underset{\mathbf{x}\in S}{\text{Minimize}} \quad f_1(\mathbf{x}) = 4x_1 + x_2 \tag{19.19h}$$



For this simple problem, we found the solution to this problem is all points in the line segment between points B and C, as shown in the figure above. The function value is $f_1(\mathbf{x}) = 4x_1 + x_2 = 10.02$. Because the optimum solution is not unique, we continue by minimizing the second objective function $f_2 = 2x_1 + x_2$ while adding the result of the first optimization problem to the constraint set:

$$\underset{\mathbf{x}\in S}{\text{Minimize:}}\ f_2(\mathbf{x}) = 2x_1 + x_2 \tag{19.19i}$$

$$\text{Subject to: } f_1(\mathbf{x}) \leq f_1(\mathbf{x}_1^*) = 10.02 \tag{19.19j}$$

Note that Eq. 19.19j is the additional constraint from the result of the first optimization. For the second optimization problem, we found the solution at point B, where $f_2 = 5.47$, while f_1 is unchanged, satisfying the constraint function defined in Eq. 19.19j. Because the solution is unique, the process stops. The solution is found at point B, where $f_1 = 10.02, f_2 = 5.47$, and $f_3 = 3.21$. Note that f_3 was not even considered in the solution process because it is the least important objective among the three. If we change the importance order of the objective functions, we will most likely reach a different solution.

The advantages of the method include that it offers a unique approach to specifying preferences, it does not require that the objective functions be normalized, and it always provides a Pareto optimal solution. A few disadvantages on this method include that it can require the solution of many single-objective problems to obtain just one solution point, and it requires that additional constraints be imposed. When there are more objective functions, the constraint set becomes large toward the end of the solution process.

19.3.3 METHODS WITH A POSTERIORI ARTICULATION OF PREFERENCES

In some cases, it is difficult for a designer to express preferences a priori. Therefore, it can be effective to allow the designer to choose from a palette of solutions. To this end, a number of methods

aim at producing almost all the Pareto optimal solutions or a representative subset of the Pareto optimal. Such methods incorporate a posteriori articulation of preferences; they are called cafeteria or generate-first-choose-later approaches (Messac and Mattson, 2002). Several methods belong to this category, such as the normal boundary intersection method (Das and Dennis, 1998) and adaptive weighted-sum (Kim and de Weck, 2006), to name a few. In addition, methods based on evolution algorithms (EA), such as the genetic algorithm to be discussed in Section 19.4, are considered a posteriori methods, although they solve the MOO problems in a much different way. In this subsection, we introduce one of the representative methods in this category, the normal boundary intersection (NBI) method.

19.3.3.1 Normal Boundary Intersection Method

The normal boundary intersection method was developed in response to deficiencies in the weighted-sum method. This method provides a means for obtaining an even distribution of Pareto optimal points, even with a nonconvex Pareto front of varying curvature. We use a two-objective problem shown in Figure 19.10 to illustrate the concept. The approach is formulated as follows:

$$\underset{\mathbf{x}\in S}{\text{Maximize: }} \beta \tag{19.20a}$$

$$\text{Subject to: } \boldsymbol{\alpha} + \beta\mathbf{n} = \mathbf{f}(\mathbf{x}) \tag{19.20b}$$

where $\boldsymbol{\alpha}$ is a point on the line segment AB, called the convex hull of individual minima (CHIM), which is also called the utopia line. Points A and B are the optimal points of the objective functions f_1 and f_2, respectively. $\mathbf{n}$ is a vector perpendicular to CHIM and pointing toward the Pareto front. The parameter β is a scalar to be maximized. Essentially, the concept of NBI is identifying a point $\boldsymbol{\alpha}$ on the CHIM and searching a Pareto point along the $\mathbf{n}$ direction by maximizing parameter β. Because the constraint equation (Eq. 19.20b) ensures an attainable design point $\mathbf{x}$ in the feasible set S, maximizing β pushes the vector $\mathbf{n}$ to eventually intersect the Pareto front and yields a Pareto solution. If $\boldsymbol{\alpha}$ points are chosen uniformly along the CHIM, a set of fairly evenly distributed Pareto points can be reasonably expected.

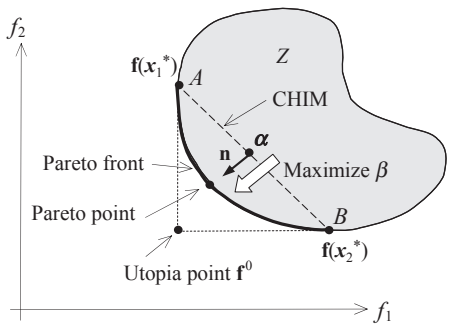


FIGURE 19.10

Geometrical illustration of the NBI method.

We use the same MOO problem of Example 19.3 to further illustrate the NBI method.

EXAMPLE 19.7

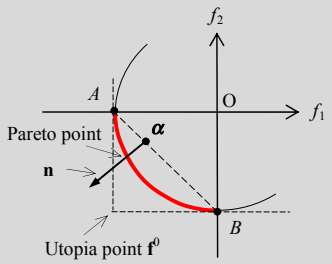
Solve the same MOO problem of Example 19.3 using the NBI method.

Solution

In this case, we recall that the utopia point is found at $\mathbf{f}^0 = (-1, -1)$, and points $A = (-1, 0)$ and $B = (0, -1)$. The CHIM line connecting points A and B is shown below. For this simple example, $\mathbf{n}$ can be found as $\mathbf{n} = [-1, -1]^T$. Therefore, using the NBI method, we are essentially solving the following single-objective optimization subproblem:

$$\underset{\mathbf{x}\in S}{\text{Maximize: }} \beta \tag{19.21a}$$

$$\text{Subject to: } [\alpha_1, \alpha_2]^T + \beta[-1, -1]^T = [x_1, x_2]^T \tag{19.21b}$$



Criterion space

The single-objective problem can be solved, for example, using MATLAB (Script 19.5). If we choose $\boldsymbol{\alpha} = [-0.7, -0.3]^T$, the solution is obtained as $x_1 = -0.8782$, $x_2 = -0.4782$, and $\beta = 0.1782$, which is on the Pareto front. Note that β is the distance between $\boldsymbol{\alpha}$ and the Pareto point found. We may pick another $\boldsymbol{\alpha}$ point to obtain another Pareto point. For example, if we choose $\alpha_1 = -0.5$ and $\alpha_2 = -0.5$, the first term in Eq. 19.21b becomes $[-0.5, -0.5]^T$, and solutions are $x_1 = -0.7071$, $x_2 = -0.7071$, and $\beta = 0.2071$.

In fact, for this simple problem, β can be solved from Eqs 19.21a and 19.21b as

$$\beta = \frac{1}{2}\left[(\alpha_1 + \alpha_2) + \sqrt{2 - (\alpha_1^2 - \alpha_2^2)}\right] \tag{19.21c}$$

which does not require solving the single-objective optimization problem of Eqs 19.21a and 19.21b. Derivation of Eq. 19.21c is left as an exercise.

For problems with more than two objectives, the NBI method can be formulated in a more general form as

$$\underset{\mathbf{x}\in S}{\text{Maximize: }} \beta \tag{19.22a}$$

$$\text{Subject to: } \boldsymbol{\Phi}\mathbf{w} + \beta\mathbf{n} = \mathbf{f}(\mathbf{x}) - \mathbf{f}^o \tag{19.22b}$$

Here, $\boldsymbol{\Phi}$ is a $q \times q$ payoff matrix with ith column composed of the vector $\mathbf{f}(\mathbf{x}_i^*) - \mathbf{f}^0$, where $\mathbf{f}(\mathbf{x}_i^*)$ is the vector of objective functions evaluated at the minimum of the ith objective function. As a result, the diagonal elements of $\boldsymbol{\Phi}$ are zeros. $\mathbf{w}$ is a vector of scalars such that $\sum_{i=1}^{q} w_i = 1$ and $w_i \geq 0 \ \forall \ i$. $\mathbf{n} = -\boldsymbol{\Phi}\mathbf{e}$, where $\mathbf{e} \in R^q$ is a column vector of ones in the criterion space. $\mathbf{n}$ is called a quasi-normal

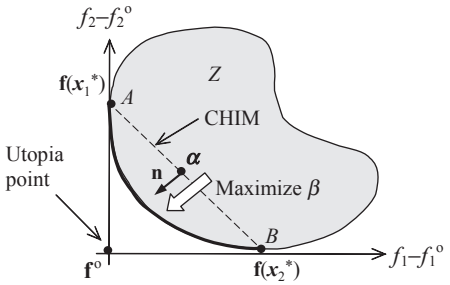


FIGURE 19.11

Geometric illustration of the NBI method for the problem defined in Eq. 19.22.

vector. Because each component of $\mathbf{\Phi}$ is positive, the negative sign ensures that $\mathbf{n}$ points toward the origin of the criterion space. $\mathbf{n}$ gives the NBI method the property that for any $\mathbf{w}$, a solution point is independent of how the objective functions are scaled. As $\mathbf{w}$ is systematically modified, the solution to Eq. 19.22 yields an even distribution of Pareto optimal points well representing the Pareto set.

We use a two-objective case shown in Figure 19.11 to illustrate the formulation geometrically. The right-hand side of Eq. 19.22b shifts the feasible criterion space Z such that the utopia point coincides with the origin of the two coordinate axes. The matrix $\mathbf{\Phi}$ for this example is constructed as

$$\mathbf{\Phi} = \left[\mathbf{f}\left(\mathbf{x}_1^*\right) - \mathbf{f}^o \;\; \mathbf{f}\left(\mathbf{x}_2^*\right) - \mathbf{f}^o\right] = \begin{bmatrix} f_1\left(\mathbf{x}_1^*\right) - f_1^o & f_1\left(\mathbf{x}_2^*\right) - f_1^o \\ f_2\left(\mathbf{x}_1^*\right) - f_2^o & f_2\left(\mathbf{x}_2^*\right) - f_2^o \end{bmatrix} = \begin{bmatrix} 0 & f_1\left(\mathbf{x}_2^*\right) - f_1^o \\ f_2\left(\mathbf{x}_1^*\right) - f_2^o & 0 \end{bmatrix} \tag{19.23}$$

The vector $\mathbf{n}$ can be written as

$$\mathbf{n} = -\mathbf{\Phi}\mathbf{e} = -\begin{bmatrix} 0 & f_1\left(\mathbf{x}_2^*\right) - f_1^o \\ f_2\left(\mathbf{x}_1^*\right) - f_2^o & 0 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = -\begin{bmatrix} f_1\left(\mathbf{x}_2^*\right) - f_1^o \\ f_2\left(\mathbf{x}_1^*\right) - f_2^o \end{bmatrix} \tag{19.24}$$

Hence, the constraint Eq. 19.22b is

$$\mathbf{\Phi}\mathbf{w} + \beta\mathbf{n} = \begin{bmatrix} w_2\left[f_1\left(\mathbf{x}_2^*\right) - f_1^o\right] \\ w_1\left[f_2\left(\mathbf{x}_1^*\right) - f_2^o\right] \end{bmatrix} - \beta\begin{bmatrix} f_1\left(\mathbf{x}_2^*\right) - f_1^o \\ f_2\left(\mathbf{x}_1^*\right) - f_2^o \end{bmatrix} = \mathbf{f}(\mathbf{x}) - \mathbf{f}^o = \begin{bmatrix} f_1(\mathbf{x}) - f_1^o \\ f_2(\mathbf{x}) - f_2^o \end{bmatrix} \tag{19.25}$$

Note that the first term on the left-hand side of Eq. 19.25 ($\mathbf{\Phi}\mathbf{w}$) is nothing but a point on the CHIM line in the criterion space with axes $f_1 - f_1^o$ and $f_2 - f_2^o$. The vector in the second term is the normal vector $\mathbf{n}$ in Eq. 19.20b. Therefore, the constraint equations in Eqs 19.22b and 19.20b are very similar. The major difference is that in Eq. 19.22b the $\boldsymbol{\alpha}$ points are generated by adjusting the vector $\mathbf{w}$. For MOO problems with more than two objectives ($q > 2$), in which the CHIM becomes an utopia-hyperplane, Eq. 19.22b is more general in determining a set of uniformly distributed $\boldsymbol{\alpha}$ points, as well as the normal vector $\mathbf{n}$. As a result, uniformly distributed Pareto solutions can be expected.

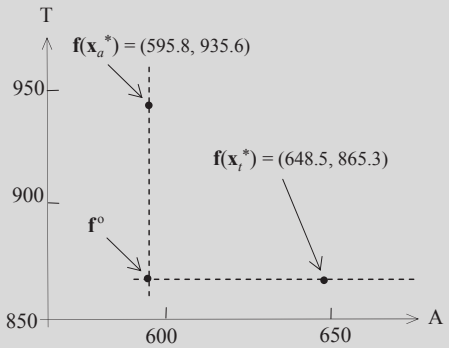
In Example 19.8, we revisit the pyramid example to illustrate more details on the NBI method formulated in Eq. 19.22.

EXAMPLE 19.8

Solve the pyramid problem using the NBI method.

Solution

From Example 19.4, we have solved the two respective single-objective optimization problems for the optimal points $\mathbf{x}_t^*$ and $\mathbf{x}_a^*$ in the design space. The objective functions at these two design points are located in the criterion space $\mathbf{f}(\mathbf{x}_t^*) = (A(\mathbf{x}_t^*), T(\mathbf{x}_t^*))$ and $\mathbf{f}(\mathbf{x}_a^*) = (A(\mathbf{x}_a^*), T(\mathbf{x}_a^*))$ in the figure shown below. Hence, the utopia point is found at $\mathbf{f}^0 = (A(\mathbf{x}_a^*), T(\mathbf{x}_t^*)) = (595.8, 865.3)$.



Using the NBI method, the constraint equation of Eq. 19.22b can be found, using Eq. 19.25, as

$$\mathbf{\Phi w} + \beta \mathbf{n} = \begin{bmatrix} w_2\left[A\left(\mathbf{x}_t^*\right) - A\left(\mathbf{x}_a^*\right)\right] \\ w_1\left[T\left(\mathbf{x}_a^*\right) - T\left(\mathbf{x}_t^*\right)\right] \end{bmatrix} - \beta \begin{bmatrix} \left[A\left(\mathbf{x}_t^*\right) - A\left(\mathbf{x}_a^*\right)\right] \\ \left[T\left(\mathbf{x}_a^*\right) - T\left(\mathbf{x}_t^*\right)\right] \end{bmatrix} = \mathbf{f}(\mathbf{x}) - \mathbf{f}^{\circ} = \begin{bmatrix} A\left(\mathbf{x}\right) - A\left(\mathbf{x}_a^*\right) \\ T\left(\mathbf{x}\right) - T\left(\mathbf{x}_t^*\right) \end{bmatrix} \tag{19.26a}$$

Plugging in the numbers, we have

$$\begin{bmatrix} w_2(648.5 - 595.8) \\ w_1(935.6 - 865.3) \end{bmatrix} - \beta \begin{bmatrix} (648.5 - 595.8) \\ (935.6 - 865.3) \end{bmatrix} = \begin{bmatrix} 2a\sqrt{\left(\frac{a}{2}\right)^2 + h^2} - 595.8 \\ \left[2a\sqrt{\left(\frac{a}{2}\right)^2 + h^2} + a^2\right] - 865.3 \end{bmatrix} \tag{19.26b}$$

Hence, using the NBI method, the problem is converted into a single-objective optimization as

$$\underset{\mathbf{x} \in S}{\text{Maximize: }} \beta \tag{19.26c}$$

$$\begin{aligned} \text{Subject to: } & w_2(648.5 - 595.8) - \beta(648.5 - 595.8) - 2a\sqrt{\left(\frac{a}{2}\right)^2 + h^2} + 595.8 = 0 \\ & w_1(935.6 - 865.3) - \beta(935.6 - 865.3) - \left[2a\sqrt{\left(\frac{a}{2}\right)^2 + h^2} + a^2\right] + 865.3 = 0 \end{aligned} \tag{19.26d}$$

where S is the feasible space defined in Eq. 19.7. The single-objective problem can be solved, for example, using MATLAB (Script 19.6).

We choose $w_1 = 0.0, 0.1, \ldots, 1.0$ (and $w_2 = 1 - w_1$) and solved for these 11 cases. The results are listed in the table next page. Because the $\boldsymbol{\alpha}$ points are selected uniformly along the CHIM by specifying a uniform increment 0.1 in w_1 from 0 to 1, the Pareto points found from NBI are more evenly distributed. In this example, however, because the Pareto front is mild in

Continued

EXAMPLE 19.8—cont'd

geometry without significant changes in curvature, the advantage of the NBI is not apparent, and the results shown in the table below are similar to those in Example 19.4, in which the weighted-sum method was employed.

	Weights		Design Point		Surface Area	Total Area	
Case No.	w_1	w_2	a	h	A	T	β
0	0.0	1.0	14.72	20.76	648.5	865.4	0.0007
1	0.1	0.9	15.08	19.78	638.6	866.1	0.0887
2	0.2	0.8	15.45	18.86	629.6	868.2	0.1586
3	0.3	0.7	15.81	18.00	621.7	871.7	0.2090
4	0.4	0.6	16.18	17.19	614.8	876.6	0.2399
5	0.5	0.5	16.55	16.43	608.9	882.8	0.2514
6	0.6	0.4	16.92	15.72	604.1	890.4	0.2434
7	0.7	0.3	17.29	15.05	600.2	899.3	0.2160
8	0.8	0.2	17.67	14.41	597.4	909.6	0.1692
9	0.9	0.1	18.05	13.82	595.6	921.3	0.1032
10	1.0	0.0	18.42	13.26	594.9	934.3	0.0180

19.3.4 METHODS WITH NO ARTICULATION OF PREFERENCE

Sometimes the designer cannot concretely define what he or she prefers. Consequently, this section describes methods that do not require any articulation of preferences. One of the simplest methods is the min–max method, formulated as

$$\underset{\mathbf{x}\in S}{\text{Minimize}} \quad \max_i \left[f_i(\mathbf{x})\right] \tag{19.27}$$

The concept and formulation are straightforward. However, implementing Eq. 19.27 is not obvious. One possible approach is to introduce a new parameter β, which is to be minimized while requiring that all objective functions are no greater than β. Mathematically, the min–max problem can be formulated as

$$\underset{\mathbf{x}\in S}{\text{Minimize: }} \beta \tag{19.28a}$$

$$\text{Subject to: } f_i(\mathbf{x}) \leq \beta, \quad i = 1, q \tag{19.28b}$$

The following example (Example 19.9) illustrates the min–max method.

EXAMPLE 19.9

Solve the pyramid problem using the min—max method.

Solution

Using the min—max method, the pyramid problem is formulated as

$$\underset{\mathbf{x}\in S}{\text{Minimize: }} \beta \tag{19.29a}$$

EXAMPLE 19.9—cont'd

$$\text{Subject to: } 2a\sqrt{\left(\frac{a}{2}\right)^2 + h^2} - \beta \leq 0$$
$$2a\sqrt{\left(\frac{a}{2}\right)^2 + h^2} + a^2 - \beta \leq 0 \tag{19.29b}$$

where S is the feasible region defined in Eq. 19.7. The single-objective problem can be solved, for example, using MATLAB (Script 19.7). The solution found is design point $\mathbf{x}_t^*$, at which $a = 14.71$, $h = 20.80$, surface area $A = 649.0$, total area $T = 865.3$, and $\beta = 865.3$. The parameter $\beta = 865.3$ is indeed the minimum because it is the minimum of the total area T. Although the solution $\mathbf{x}_t^*$ does not minimize surface area A, reducing A cannot be done without increasing the total area T, in which β is no longer the minimum.

19.3.5 MULTIOBJECTIVE GENETIC ALGORITHMS*

The solution techniques we discussed so far are mainly searching a single best solution representing the best compromise given the information from the designer. These techniques are called aggregating approaches because they combine (or "aggregate") all the objectives into a single one. Such techniques approximate the Pareto front by repeating the solution process by adjusting parameters that redefine the search, such as the weights in the weighted-sum method.

In contrast to these techniques, methods based on evolution algorithms, such as multiobjective genetic algorithms, support direct generation of the Pareto front by simultaneously optimizing the individual objectives. As in the single-objective optimization problem discussed in Chapter 17, genetic algorithms emulate the biological evolution process. A population of individuals representing different solutions is evolving to find the optimal solutions. The fittest individuals are chosen, with mutation and crossover operations applied, thus yielding a new generation (offspring), as discussed in Section 17.7.1.

Evolutionary algorithms seem particularly suitable to solving multiobjective optimization problems because they deal simultaneously with a set of possible solutions (or a population). This allows designers to find several members of the Pareto optimal set in a single run of the algorithm, instead of having to perform a series of separate runs. Additionally, genetic algorithms are less susceptible to the shape or continuity of the Pareto front (e.g., they can easily deal with discontinuous or concave Pareto fronts), which are the two issues identified in the classical approaches discussed so far in this chapter.

In general, for multiobjective problems, genetic algorithms have the advantage of evaluating multiple potential solutions in a single iteration. They offer greater flexibility for the designer, mainly in cases where no a priori information is available, as is the case for most real-life multiobjective problems. However, the challenge is how to guide the search toward the Pareto optimal set, and how to maintain a diverse population in order to prevent premature convergence. In addition, as discussed in Chapter 17, genetic algorithms require a large amount of function evaluations. For complex problems that require significant computation time for function evaluations, such as problems involving nonlinear finite element analysis, methods other than genetic algorithms are more practical.

One of the first treatments of multiobjective genetic algorithms, called the vector-evaluated genetic algorithm (VEGA), was presented by Schaffer (1985). Although this method was surpassed by others soon after it was proposed, it has provided a foundation for later development. Many prominent

algorithms have been developed in the past three decades, in which ranking was explicitly used in order to determine the probability of replication of an individual. Such methods are so-called Pareto-based approaches.

In this section, we first provide a general discussion on the Pareto-based approaches. Thereafter, we zoom in to one of the most popular Pareto-based approaches, called the nondominated sorting genetic algorithm (NSGA). NSGA was proposed by Srinivas and Deb (1994) and revised to NSGA II (Deb et al., 2002), with significant improvements on the efficiency of the sorting algorithm as well as on its niche technique. We also present an example of NSGA II implementation in MATLAB, followed by the pyramid example for illustration. Note that some aspects of the method presented are shared by many other multiobjective genetic algorithms.

19.3.5.1 Pareto-Based Approaches

Pareto-based approaches incorporate three major elements—ranking, niching mechanism, and elitist strategy—in addition to the common mutation and crossover techniques seen in genetic algorithms. Ranking determines the probability of replication of an individual. The basic idea is to find the set of nondominated individuals in the population. These are assigned the highest rank and eliminated from further contention. The process is then repeated with the remaining individuals until the entire population is ranked and assigned a fitness value. In conjunction with Pareto-based fitness assignment, a niching mechanism is used to prevent the algorithm from converging to a single region of the Pareto front. The elitist strategy provides a means for ensuring that Pareto optimal solutions are not lost. Mutation and crossover operations are then performed to create the next generation of individuals.

19.3.5.1.1 Ranking

A simple and efficient method assumes that the fitness value of an individual is proportional to the number of other individuals it dominates, as illustrated in Figure 19.12. In Figure 19.12(a), point 2 dominates six other individuals (points 3, 5, 6, 7, 9, and 10) because the objective function values at this point are less than those of the six points it dominates. Therefore, point 2 is assigned a higher

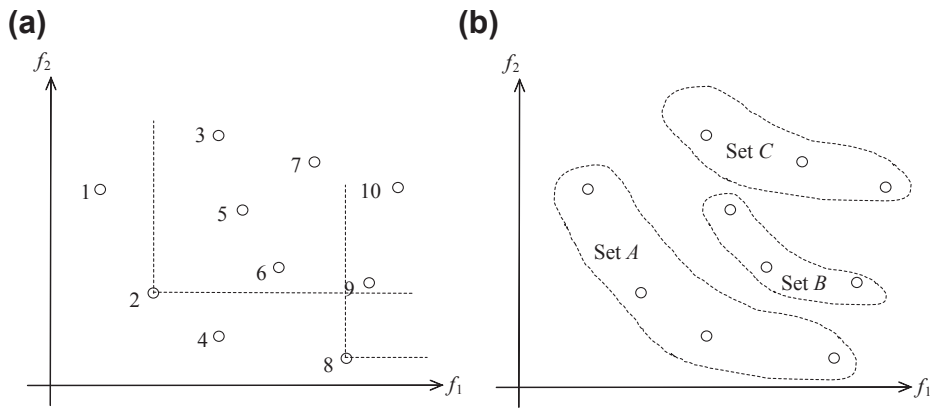


FIGURE 19.12

Illustration of fitness assignments: (a) ranking and (b) fitness computation for NSGA.

fitness, and hence the probability of selecting this point is higher than, for example point 8, which only dominates two individuals (points 9 and 10).

Another version is the nondominated sorting genetic algorithm, which uses a layered classification technique, as illustrated in Figure 19.12(b). In Figure 19.12(b), a layered classification technique is used whereby the population is incrementally sorted using Pareto dominance. Individuals in set A are assigned the same fitness value, which is higher than the fitness of individuals in set B, which in turn are superior to individuals in set C. All nondominated individuals are assigned the same fitness value. The process is repeated for the remainder of the population, with a progressively lower fitness value assigned to the nondominated individuals.

19.3.5.1.2 Niche techniques

A niche in genetic algorithms is a group of points that are close to each other, typically in the criterion space. Niche techniques (also called niche schemes, niche mechanism, or the niche-formation method) are methods for ensuring that a set of designs does not converge to a niche. Thus, these techniques foster an even spread of points in the criterion space. Genetic multiobjective algorithms tend to create a limited number of niches; they converge to or cluster around a limited set of Pareto optimal points. This phenomenon is known as genetic or population drift; niche techniques force the development of multiple niches while limiting the growth of any single niche.

Fitness sharing is a common niche technique. The basic idea of which is to penalize the fitness of points in crowded areas, thus reducing the probability of their survival for the next iteration. The fitness of a given point is divided by a constant that is proportional to the number of other points within a specified distance in the criterion space, thus reducing its fitness value and lowering its chance to survive for the next iteration. In this way, the fitness of all points in a niche is shared in some sense—thus the term "fitness sharing."

19.3.5.1.3 Elitist strategy

Elitist strategy provides a means for ensuring that Pareto optimal solutions are not lost. It functions independently of the ranking scheme. Two sets of solutions are stored: a current population and a tentative set of nondominated solutions, which is an approximate Pareto optimal set. In each iteration, all points in the current population that are not dominated by any points in the tentative set are added to the tentative set. Then, the dominated points in the tentative set are discarded. After crossover and mutation operations are applied, a user-specified number of points from the tentative set are reintroduced into the current population. These are called elite points. In addition, q points with the best values for each objective function can be regarded as elite points and preserved for the next generation. Recall that q is the number of objective functions.

19.3.5.2 Nondominated Sorting Genetic Algorithm II

With the understanding of the basic elements in the GA-based approaches for MOO problems, we now discuss NSGA II.

In NSGA II, before selection is performed, the population is ranked on the basis of nondomination: all nondominated individuals are classified into one category to provide an equal reproductive potential for these individuals. Because the individuals in the first front have the maximum fitness value, they always get more copies than the rest of the population when the selection is carried out. Additionally, the NSGA II estimates the density of solutions surrounding a particular solution in the population by

computing the average distance of two points on either side of this point along each of the objectives of the problem. This value is called the crowding distance. During selection, NSGA II uses a crowded-comparison scheme that takes into consideration both the nondomination rank of an individual in the population and its crowding distance. The nondominated solutions are preferred over dominated solutions; however, between two solutions with the same nondomination rank, the one that resides in the less crowded region is preferred. The NSGA II uses the elitist mechanism that consists of combining the best parents with the best offspring obtained. More details are described in the next section.

19.3.5.2.1 Nondominated sorting

One of the major computation efforts in performing the nondominated sorting involves comparisons. Each solution can be compared with every other solution in the population to find out if it is dominated. This requires $q\times(N-1)$ comparisons for each solution, where q is the number of objectives and N is the population size. The first round of sorting involves $\sum_{i=1}^{N-1} q(N-i) = \frac{1}{2}qN(N-1)$ comparisons, which is in the order of $O(qN^2)$.

The results of the sorting are usually stored in an $N\times N$ matrix, in which each term indicates the dominance relation between two individuals in the population. In general, four scenarios need to be considered in terms of the feasibilities of the two individuals x and y being compared. First, if both x and y are feasible, then objectives at x and y are compared. If x dominates y, then the domination matrix $\text{donMat}(x, y) = 1$; else, if y dominates x, then $\text{donMat}(x, y) = -1$. Second, if x is feasible and y is infeasible, then $\text{donMat}(x, y) = 1$. Third, if x is infeasible and y is feasible, then $\text{donMat}(x, y) = -1$. Fourth, if both x and y are infeasible, then the constraint violations of x and y are compared. If violation at x is less than that of y, then $\text{donMat}(x, y) = 1$. Otherwise, $\text{donMat}(x, y) = -1$.

To further illustrate the idea, we use the example of population size $N = 10$ shown in Figure 19.12(a). To simplify the discussion, we assume a nonconstrained problem. The matrix is set to zero initially. We pick point 2 to illustrate the process. As shown in Figure 19.12(a), point 2 dominates points 3, 5, 6, 7, 9, and 10. Therefore, in the following matrix, entries (2, 3), (2, 5), (2, 6), (2, 7), (2, 9), and (2, 10) are set to 1. The entries (3, 2), (5, 2), (6, 2), (7, 2), (9, 2), and (10, 2) are set to −1. By going over the comparisons, the domination matrix can be constructed, as shown in Figure 19.13.

As proposed by Deb et al. (2002), for each individual we calculate two entities: (1) domination count n_p, the number of solutions that dominate the solution p, $p = 1, N$; and (2) D_p, a set of individuals

	1	2	3	4	5	6	7	8	9	10
1	0	0	−1	0	0	0	−1	0	0	−1
2	0	0	−1	0	−1	−1	−1	0	−1	−1
3	1	1	0	1	0	0	0	0	0	0
4	0	0	−1	0	−1	−1	−1	0	−1	−1
5	0	1	0	1	0	0	−1	0	0	−1
6	0	1	0	1	0	0	−1	0	0	−1
7	1	1	0	1	1	1	0	0	0	0
8	0	0	0	0	0	0	0	0	−1	−1
9	0	1	0	1	0	0	0	1	0	−1
10	1	1	0	1	1	1	0	1	1	0

FIGURE 19.13

Domination matrix for the example shown in Figure 19.12(a).

Table 19.2 Domination Count n_p and the Set D_p Derived from the Domination Matrix

Solution Point p	Solution p Dominates (D_p)	Solutions Dominate p	n_p
1	3, 7, 10	None	0
2	3, 5, 6, 7, 9, 10	None	0
3	None	1, 2, 4	3
4	5, 6, 7, 9, 10	None	0
5	7, 10	2, 4	2
6	7, 10	2, 4	2
7	None	1, 2, 4, 5, 6	5
8	9, 10	None	0
9	10	2, 4, 8	3
10	None	1, 2, 4, 5, 6, 8, 9	7

that the solution dominates. The count n_p can be found by adding the number of -1 of the pth column in the matrix. The set D_p can be created by collecting individuals with -1 along the pth row in the matrix. For example, for point 2, $n_2 = 0$ and $D_2 = \{3, 5, 6, 7, 9, 10\}$. By checking the columns and rows of the domination matrix, Table 19.2 can be constructed.

All solutions in the first nondominated front will have their domination count as zero ($n_p = 0$). From Table 19.2, the first nondominated front is identified as $ND_1 = \{1, 2, 4, 8\}$. Now, for each solution in ND_1, we visit each member of its set D_p and reduce its domination count by 1. For example, for the second member of ND_1 (point 2), the counts of its members in D_2 become respectively, $n_3 = 3 - 1 = 2$, $n_5 = 1$, $n_6 = 1$, $n_7 = 4$, $n_9 = 2$, and $n_{10} = 6$. In doing so, if for any member the domination count becomes zero (if none is 0, we pick the ones with the lowest integer), these members belong to the second nondominated front ND_2. In this example, solutions 5 and 6 of D_2 have the lowest count 1. After repeating the same process for the remaining solutions 1, 4, and 8 in ND_1, the second nondominated front is $ND_2 = \{5, 6\}$. Now, the above procedure is continued with each member of ND_2, and the third front is identified as $ND_3 = \{7\}$. This process continues until all fronts are identified. Note that some individuals may not get counted, for example, 3, 9, and 10 in this case, which are dominated points. It is important that the first few nondominated fronts are identified accurately. In this example, the first two fronts, ND_1 and ND_2, are accurately identified as can be seen in Figure 19.12(a).

19.3.5.2.2 Niche technique for diversity preservation

NSGA II employs a different niche technique to preserve the diversity of the Pareto solutions. This technique involves two factors: density-estimation metric and crowded-comparison operator.

Density estimation calculates an estimate of the density of points surrounding a particular solution in the population. The estimate calculates the average distance of two points on either side of this point along each of the objectives. This quantity serves as an estimate of the perimeter of the cuboid formed by using the nearest neighbors as the vertices (this is called the crowding distance). For example, in Figure 19.14, the crowding distance of the ith solution in its front (marked with solid dots) is the average side length of the cuboid (shown with a dashed box). The crowding-distance

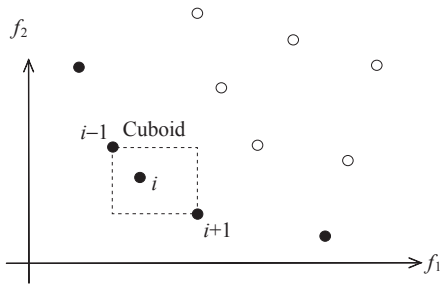


FIGURE 19.14

Crowding-distance calculation. Points marked in filled circles are solutions of the same nondominated front.

computation requires sorting the population according to each objective function value in ascending order of magnitude. Thereafter, for each objective function, the boundary solutions (solutions with the smallest and largest function values) are assigned an infinite distance value. All other intermediate solutions are assigned a distance value equal to the absolute normalized difference in the function values of two adjacent solutions. This calculation is continued with other objective functions. The overall crowding-distance value is calculated as the sum of individual distance values corresponding to each objective. Note that each objective function is normalized before calculating the crowding distance.

The crowded-comparison operator ($\prec$) guides the selection process at the various stages of the algorithm toward a uniformly spread-out Pareto optimal front. Assume that every individual in the population has two attributes:

1. Nondomination rank (i_{rank})
2. Crowding distance (i_{distance}).

We now define a partial order as

$i \prec j$ if ($i_{\text{rank}} < j_{\text{rank}}$) or if ($i_{\text{rank}} = j_{\text{rank}}$, $i_{\text{distance}} > j_{\text{distance}}$).

That is, between two solutions with different nondomination ranks, we prefer the solution with the lower (better) rank. Otherwise, if both solutions belong to the same front, then we prefer the solution that is located in a lesser crowded region.

At the end of nondominated sorting, the individuals of the population of size N are sorted based on their rank. For the individuals of same rank, they are sorted by crowding distances.

19.3.5.2.3 Overall process

The overall NSGA II process consists of the creation of initial population and design iterations. Initially, a random parent population P_0 of size N is created. Function evaluations of objectives and constraints are carried out for each of the N individuals. The population is sorted based on the nondomination and crowding distance, as discussed above.

At first, the usual binary tournament selection, recombination, and mutation operators are used to create an offspring population Q_0 of size N. The procedure is as follows. Function evaluations and nondominated (and crowding distance) sorting are performed. Then, a binary tournament selection is carried out to select the best N individuals. These N individuals go through crossover and mutation to

A. Assume that the current generation P is [①, ②, ③, ④, ⑤, ⑥], so the size of the population is $N = 6$.

B. Assume that ② > ③ > ① > ⑥ > ④ > ⑤, where ② > ③ means the individual ② is better than ③. The comparison between each two individuals is based on their ranks (if ranks are the same, then compare crowding distance).

C. The next step is to create a new population Q_0 of size N. The procedure is shown below

Create $2N$ (in this case 12) random integers between 1 and 6, for example:

[⑤, ④, ①, ③, ②, ②, ⑥, ⑤, ①, ④, ③ ⑤]

Carry out binary tournament selection, crossover, and mutation:

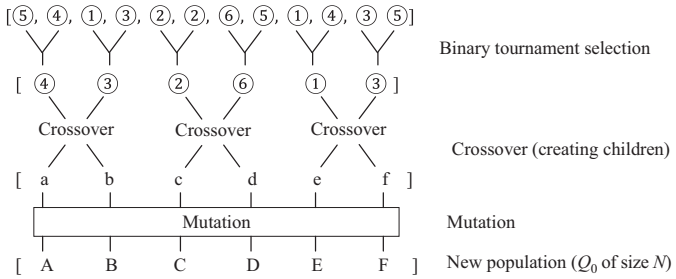


FIGURE 19.15

An example of the selection of new population of size $N = 6$.

generate a new population Q_0 of size N. An example of these steps is shown in Figure 19.15 for illustration, in which $N = 6$ is assumed.

Because elitism is introduced by comparing the current population with the previously found best nondominated solutions, the procedure is different after the initial generation. We describe the ith generation of the algorithm. First, a combined population $R_i = P_i \cup Q_i$ is formed. The population is of size $2N$. Then, the population is sorted according to nondomination (and crowding distance). Because all previous and current population members are included in R_i, elitism is ensured. Now, solutions belonging to the best nondominated set F_1 are the best solutions in the combined population (see Figure 19.16) and must be emphasized more than any other solution in the combined population. If the size of F_1 is smaller than N, we choose all members of the set F_1 for the new population P_{i+1}. The remaining members of the population P_{i+1} are chosen from subsequent nondominated fronts in the order of their ranking. Thus, solutions from the set F_2 are chosen next, followed by solutions from the set F_3, and so on. This procedure is continued until no more sets can be accommodated. Say that the set F_ℓ is the last nondominated set beyond which no other set can be accommodated. In general, the count of solutions in all sets from F_1 to F_ℓ would be larger than the population size. To choose exactly N population members, we sort the solutions of the last front using the crowded-comparison operator in descending order and choose the best solutions needed to fill all population slots. This procedure is also shown in Figure 19.16. The new population P_{i+1} of size N is now used for selection, crossover, and mutation to create a new population Q_{i+1} of size N.

19.3.5.3 Sample MATLAB Implementation

In this section, we review a sample implementation of NSGA II written in MATLAB, and then we use this code to solve the pyramid example for illustration. The code is available for download at either its

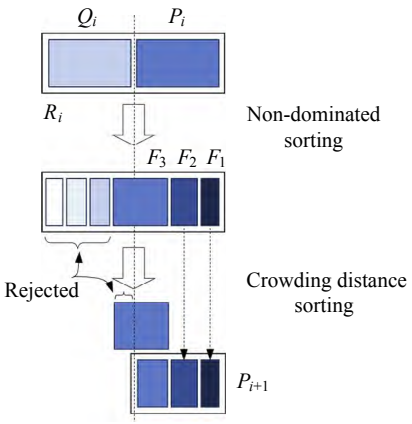


FIGURE 19.16

Procedure of selecting a new population P_{i+1}.

original site (www.mathworks.com/matlabcentral/fileexchange/31166-ngpm-a-nsga-ii-program-in-matlab-v1-4) or from the book's companion site. The zipped file consists of a set of MATLAB files and three folders. The doc folder includes a license file and a user manual. The other two folders, TP_NSGA2 and TP_R-NSGA2, include test problems that come with the code. Readers may download the code to go over some of the exercises at the end of this chapter.

There are two MATLAB files that users will have to create: one contains the input parameters, such as the number of objectives, whereas the other defines the objective and constraint functions. Once the two MATLAB files are created, we run the file with input parameters in MATLAB. The code searches for solutions at the Pareto front following the NSGA II algorithm discussed above. During the solution process, the code shows a list of generations, plots the solutions, and provides a summary on the status of the optimization, as shown in Figure 19.17.

To solve the pyramid example using the code, we first create the two needed files and name them, respectively, PYRAMID.m and PYRAMID_objfun.m. The contents of these two files are shown in Figure 19.18. Note that the name of the function needs to match that defined in PYRAMID.m. In this case, the name of the function must be options.objfun = @PYRAMID_objfun, as circled in red in Figure 19.18(a).

As shown in Figure 19.18(a), the inputs of the pyramid example include the following:

- Population size: 200
- Number of generations: 500
- Number of objective functions: 2
- Number of design variables: 2
- Number of constraints: 1
- Lower bounds of design variables: 0 for the two design variables
- Upper bounds of design variables: 30 for the two design variables

The MATLAB script PYRAMID.m is called to start the optimization process. Results of a number of iterations are graphed in Figure 19.19, in which the red dotted line represents the Pareto front. As

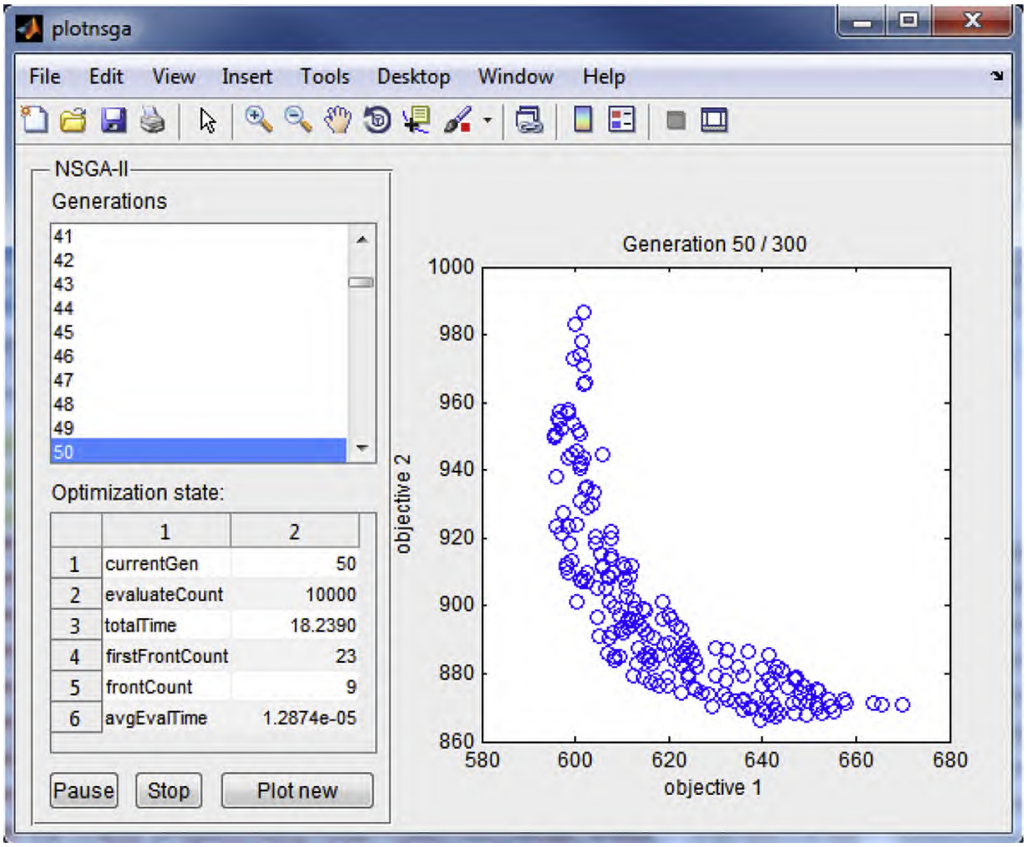


FIGURE 19.17

Screen capture of an intermediate iteration in MATLAB.

(a)

```
%**************************************************************************
% Test Problem : 'PYRAMID'
%**************************************************************************

options = nsgaopt();                    % create default options structure
options.popsize = 200;                  % populaion size
options.maxGen  = 500;                  % max generation

options.numObj = 2;                     % number of objectives
options.numVar = 2;                     % number of design variables
options.numCons = 1;                    % number of constraints
options.lb = [0 0];                     % lower bound of x
options.ub = [30 30];                   % upper bound of x
options.objfun = @PYRAMID_objfun;       % objective function handle
options.plotInterval = 5;               % interval between two calls of "plotnsga".
result = nsga2(options);                % begin the optimization!
```

(b)

```
function [y, cons] = PYRAMID_objfun(x)
% Objective function : Test problem 'PYRAMID'.
%y(1): S      y(2): T
%x(1): a      x(2): h
%**************************************************

y = [0,0];
cons = 0;

y(1) = 2*x(1)*sqrt(x(1)^2/4+x(2)^2);
y(2) = 2*x(1)*sqrt(x(1)^2/4+x(2)^2)+x(1)^2;

% calculate the constraint violations
c = x(1)^2*x(2)/3 - 1500;
if(c<0)
    cons(1) = abs(c);
end
```

FIGURE 19.18

Contents of the two user-created files: (a) file: PYRAMID and (2) file: PYRAMID_objfun.

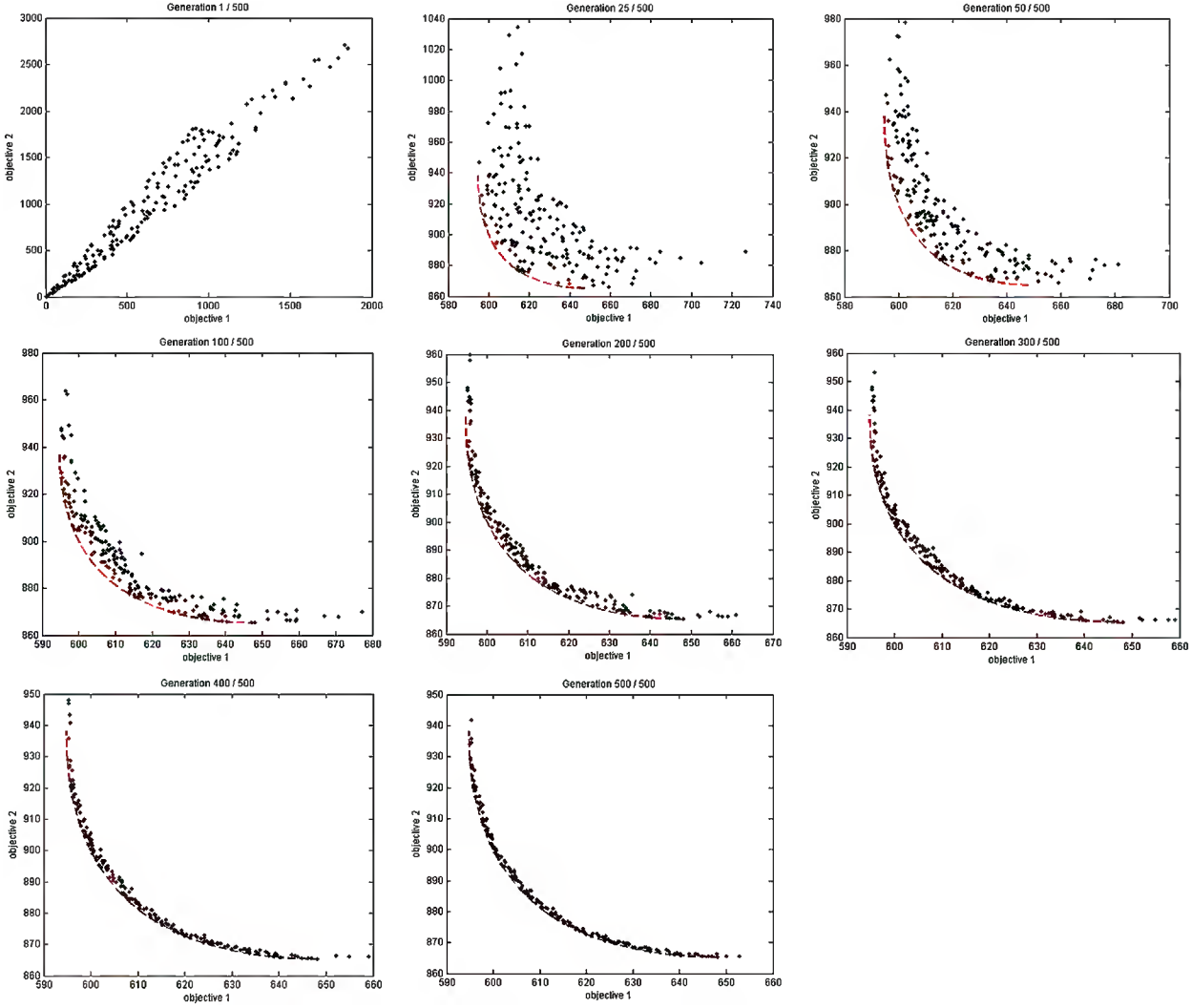


FIGURE 19.19

Result graphs of the pyramid example at selected iterations.

seen in Figure 19.19, the solutions converge rapidly to the Pareto front at the 200th iteration. At the last iteration (500), the Pareto front is well captured.

In general, the genetic algorithms successfully address the limitations of the classical approaches when generating the Pareto front. Because they allow concurrent exploration of different points of the Pareto front, they can generate multiple solutions in a single run. The optimization can be performed without a priori information about objectives' relative importance. These techniques can handle ill-posed problems with incommensurable or mixed-type objectives. They are not susceptible to the shape of the Pareto front. Their main drawback is performance degradation as the number of objectives increases. Furthermore, they require additional parameters such as crossover fraction and mutation fraction, which need to be tweaked to improve the performance of the algorithms.

19.4 DECISION-BASED DESIGN

In general, engineering design is viewed as a problem-solving process. Such as in this chapter, a design problem is formulated mathematically as optimization problem and solved by meeting functional requirements subject to constraints at minimum objectives.

As mentioned in Chapter 16, engineering design is increasingly recognized as a decision-making process. From a product development perspective, design involves a series of decisions—some of which may be made sequentially and others that must be made concurrently. The term decision-based design (DBD) was introduced in 1990s. A formal definition introduced in Hazelrigg (1998) states that decision-based design is a normative approach that prescribes a methodology to make unambiguous design alternative selections under uncertainty and risk wherein the design is optimized in terms of the expected utility.

Uncertainty and risk involved in decision making, as well as two prominent decision theories, utility theory and game theory, were introduced in Chapter 16. In addition, we presented examples to illustrate the application of these theories to engineering design. These examples involve multiple objectives. In this section, we revisit these examples and theories. Although we do not advocate bringing decision-based design into the framework of multiobjective optimization, we compare the methods of MOO and DBD for dealing with multiobjective problems. We provide closure in treating design as a problem-solving and decision-making process by bringing the decision theories discussed in Chapter 16 into the context of multiobjective optimization.

19.4.1 UTILITY THEORY AS A DESIGN TOOL: CANTILEVER BEAM EXAMPLE

In Chapter 16, the cantilever beam example shown in Figure 19.20 was employed to illustrate the application of utility theory as a design tool. Both constrained and nonconstrained problems were solved. In this section, we bring back the results of the constrained problem for discussion.

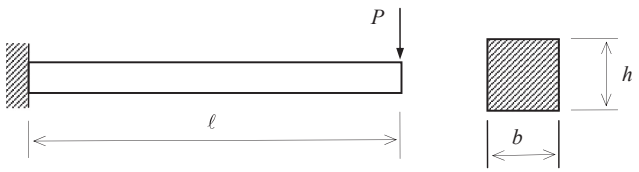


FIGURE 19.20

Cantilever beam example.

The constrained problem involves minimizing weight w and vertical displacement z and subject to bending stress σ. Mathematically, the MOO problem is formulated as

$$\text{Minimize: } f_w = w(b,h) = \rho g b h \ell, \quad \text{and} \quad f_z = z(b,h) = z = \frac{4P\ell^3}{Ebh^3} \tag{19.30a}$$

$$\text{Subject to: } g_\sigma = \sigma(b,h) - S_y \leq 0 \tag{19.30b}$$

$$10 \leq b \leq 60 \text{ mm}, \text{ and } 20 \leq h \leq 80 \text{ mm} \tag{19.30c}$$

where $\sigma = \frac{6P\ell}{bh^2}$ and the yield strength is $S_y = 27.6$ MPa.

As discussed in Chapter 16, the MOO problem was converted into a single-objective nonconstrained optimization problem, defined as

$$\text{Maximize: } u(w,z,\sigma) = [k_c + (1-k_c)u(w,z)]u(\sigma) \tag{19.31}$$

in which $u(w, z)$ is the multiattribute utility (MAU) function defined in Eq. 16.56. We assume $0 \leq k_w \leq 1$, $0 \leq k_z \leq 1$, and $k_w + k_z = 1$ (recall that k_w and k_z are the scaling constants representing the designer's preference between attributes w and z). Hence, the multiplicative MAU is reduced to an additive MAU, as in Eq. 19.32:

$$u(w,z) = k_w u_w + k_z u_z \tag{19.32}$$

which is similar to the weighted-sum method for solving MOO problems.

Also, in Eq. 19.31, the utility function of the stress constraint is defined in Eq. 19.33 as

$$u(\sigma) = \frac{1}{1 + e^{\frac{\sigma_{0.5} - \sigma}{s}}} \tag{19.33}$$

in which s is the slope of the utility function $u(\sigma)$ at $u = 0.5$, and we chose $\sigma_{0.5} = S_y = 27.6$ MPa and $s = -0.01$ in the example (see Section 16.6.1.2).

We restate the results of the five cases below in Table 19.3.

We are now solving the MOO problem defined in Eq. 19.30. We plot the Pareto front using the generative method similar to that of the pyramid problem (MATLAB Script 19.8). The boundary points of the front are identified as $\mathbf{x}_w^* = (10.01, 65.95)$ and $\mathbf{x}_z^* = (59.97, 79.94)$, as shown in Figure 19.21. At the points, the objective functions are $\mathrm{f}(\mathbf{x}_w^*) = (3.497 \text{ N}, 0.1615 \text{ mm})$ and $\mathrm{f}(\mathbf{x}_z^*) = (25.39 \text{ N}, 0.01514 \text{ mm})$, respectively, as shown in the zoomed-in figures A and B in Figure 19.21.

Table 19.3 Results Comparison for the Constrained Design Problems

	Problem Setup				Results					
Case No.	***kw***	***kz***	***rw***	***rz***	***b* (mm)**	***h* (mm)**	***w* (N)**	***z* (mm)**	***σ* (MPa)**	***u***
Case 4	0.5	0.5	0	0	10	66.1	3.50	0.161	27.46	0.950
Case 5a	0.7	0.3	0	0	10	66.1	3.50	0.161	27.46	0.938
Case 5b	0.3	0.7	0	0	10	71.3	3.78	0.128	23.60	0.962
Case 6a	0.5	0.5	−2	0	10	66.1	3.50	0.161	27.46	0.906
Case 6b	0.5	0.5	2	0	10	72.8	3.85	0.120	22.64	0.977

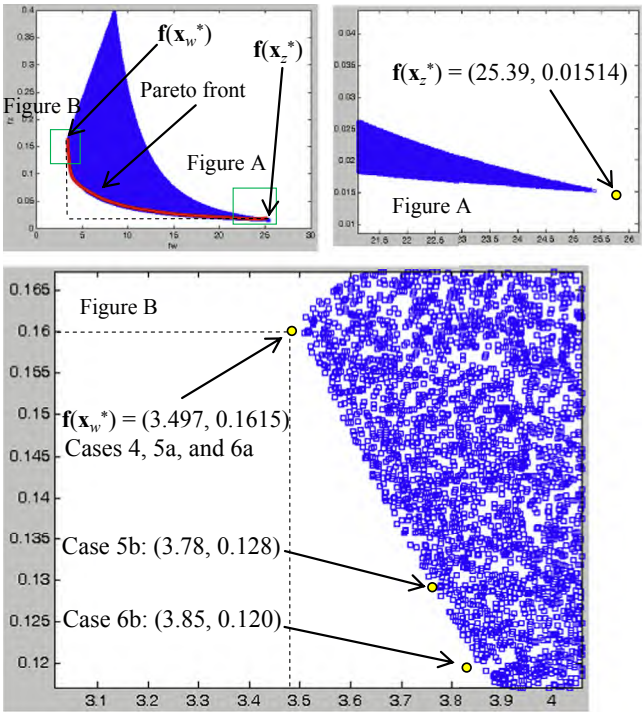


FIGURE 19.21

Pareto solutions of the beam example in criterion space.

The results obtained in Chapter 16 (see Table 19.3) show that the base case (Case 4, $k_w = k_z = 0.5$) is identical to $\mathbf{f}(\mathbf{x}_w^*)$, which is a boundary point of the Pareto front shown in Figure 19.21. In Case 5a, k_w increases to 0.7, implying that the preference is given to weight; the design is identical to that of the base case. Adjusting the preference to weight did not alter the design. This is because that there is no more room to further minimize weight without violating the stress constraint. In Case 6a, r_w is reduced to -2, indicating that the designer is risk prone to the weight attribute; the design is identical to that of the base case. Adjusting the risk attitude toward the weight did not alter the design due to the same reason. In Case 5b, k_w decreases to 0.3, implying that the preference is given to displacement; the design moves away from $\mathbf{x}_w^*$, as shown in the zoomed-in figure B of Figure 19.21. A similar result is found in Case 6b. However, all cases led to designs that are cluster to $\mathbf{x}_w^*$. The results indicate that there is a large portion of the Pareto front not being explored.

As seen in the beam example, the approach of using the utility theory as a design tool converts the MOO into a single-objective nonconstrained problem by incorporating designer's preference and risk attitude. From a MOO perspective, this approach is similar to methods with a priori articulation of preferences discussed in Section 19.3.2 in the context of conventional MOO solution techniques.

19.4.2 GAME THEORY AS A DESIGN TOOL: PRESSURE VESSEL EXAMPLE

The pressure vessel shown in Figure 19.22 was investigated using game theory as a design tool in Chapter 16. The same design problem can also be formulated as a MOO problem:

$$\text{Minimize: } W(R,t) = \gamma\left[\pi(R+t)^2(\ell+2t) - \pi R^2\ell\right] \tag{19.34a}$$

$$\text{Maximize: } V(R,t) = \pi R^2\ell \ (\text{or Minimize} - V) \tag{19.34b}$$

$$\text{Subject to: } t + R \leq 40 \tag{19.34c}$$

$$0.5 \leq t \leq 6 \text{ in.} \tag{19.34d}$$

$$5t \leq R \leq 8t \tag{19.34e}$$

$$2.5 < R \leq 5.55 \text{ in.} \tag{19.34f}$$

We plot the Pareto front using the generative method (MATLAB Script 19.9). Note that Eq. 19.34b was converted to minimize $-V$ in the MATLAB implementation. The boundary points of the Pareto front are points A, B, and C shown in Figure 19.23, in which $\mathbf{f}(A) = (f_W(2.5,0.5), f_V(2.5,0.5)) = (252.5, -1963)$, $\mathbf{f}(B) = (f_W(4,0.5), f_V(4,0.5)) = (395.9, -5027)$, and $\mathbf{f}(C) = (f_W(35.55,4.44), f_V(35.55,4.44)) = (42{,}440, -39{,}700)$.

Recall the solutions of Section 16.6.2, where we first found that the Nash solution in criterion space is curve BC, which is a subset of the Pareto front. In addition, point B is the solution to the sequential game, with Player W as the leader; point C is the solution to the sequential game with Player V as the leader. Both points are Pareto solutions.

As seen in the pressure vessel example, the approach of using game theory as a design tool converts a MOO problem into a series of single-objective problems by considering the objectives of the individual designers. The single-objective problems are then solved sequentially. The basic concept of handling MOO using game theory is different from those discussed in Section 19.3. In the former, multiple designers are making decisions. In the latter, a MOO problem is solved by a single designer. Practically, on many occasions, design decisions are made by individual groups or designers. In that regard, game theory as design tools may be more general and suitable for solving complex design problems of distributed design teams.

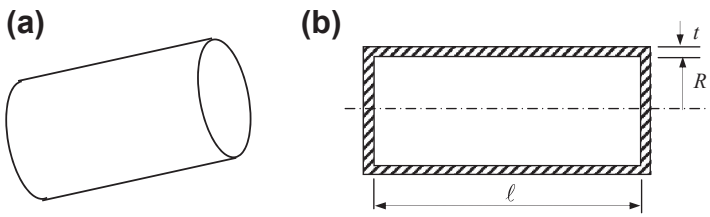


FIGURE 19.22

Cylindrical pressure vessel: (a) isometric view and (b) section view with dimensions.

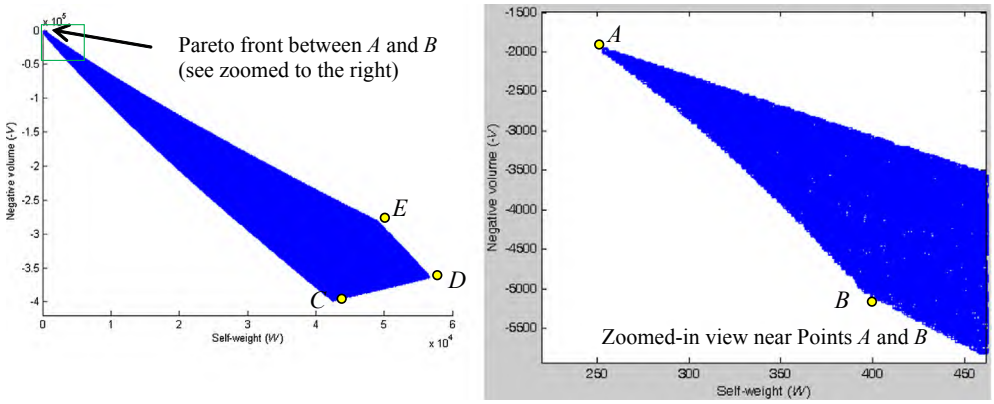


FIGURE 19.23

Pareto solutions of the beam example in criterion space.

19.5 SOFTWARE TOOLS

This section presents a general overview of academically and commercially available multiobjective optimization software. Instead of introducing detailed software capabilities, we offer brief descriptions for software tools from both categories. For readers who are selecting software for conducting optimization tasks, you are encouraged to carry out further investigations and hands-on evaluations to find a candidate software that best suits your needs. In line with the theme of this chapter, multicriteria decision-making software designed for choosing among discrete alternatives although many are readily available, will not be discussed in this section.

19.5.1 ACADEMIC CODES

A number of multiobjective optimization tools developed by universities and research laboratories are available. These software tools are usually compact and free for download from their websites. Most of them are essentially collections of multiobjective optimization algorithms implemented based on different solution techniques. For example, jMetal (jmetal.sourceforge.net) is a Java-based multiobjective optimizer; PISA (www.tik.ee.ethz.ch/pisa) is a C-based framework; interalg (openopt.org/interalg) and PaGMO/PyGMO (sourceforge.net/projects/pagmo) are programmed in Python; BENSLOVE (ito.mathematik.uni-halle.de/∼loehne/index_en_dl.php) is implemented in MATLAB; and NIMBUS (www.nimbus.it.jyu.fi) is a web-based optimization system. The capabilities of these academic tools vary from one to another due to the fact that different optimization algorithms are used. While most of them are general-purpose software, some are developed to deal with a specific group of problems; an example is the linear vector optimizer BENSLOVE, which can be used to solve only MOLP problems.

The advantage of academic codes is that they offer in general more choices of optimization techniques than commercial software packages; for example, more than 30 optimization algorithms (including single-objective and multiobjective) of different categories (gradient-based methods,

genetic algorithms, etc.) are currently available in jMetal. Moreover, the algorithm libraries in most academic software programs can be constantly updated and expanded by incorporating new modules or optimization codes contributed by users. However, because most of such software tools are intended for academic use in support of testing and research, they are usually neither well-maintained nor advertised. As a result, they tend to be less flexible in coupling with third-party engineering software (such as CAD and CAE systems), and in most cases can only be used for solving simple optimization problems with known explicit expressions of the objective and constraint functions. In addition, these academic optimization tools often require the knowledge of certain programming languages (such as MATLAB), and their graphical user interfaces, if any, are not as user-friendly as those of commercial software.

A well-known multiobjective optimization tool for teaching and academic purposes is the NIMBUS system, developed by University of Jyväskylä. NIMBUS is a free web-based interactive tool that is capable of handling both differentiable and nondifferentiable single-objective and multi-objective optimization problems subject to nonlinear and linear constraints. The central idea of NIMBUS is to first find one Pareto optimum, and then generate additional solutions on the Pareto front based on the inputs from the user, including which of the objectives should be optimized most and what the limits of the objectives are. In other words, the optimization process is directed by the user. In NIMBUS, this interactive process is called a classification, and the optimal solutions obtained in this way effectively reflect the user's preferences.

The NIMBUS system is intuitive and easy for a novice to use. Because it operates via the Internet, no software needs to be downloaded. A simple tutorial example is available at www.nimbus.it.jyu.fi/N4/tutorial/index.html. Now, we use the pyramid example to demonstrate the basic capabilities of NIMBUS. As shown in Figure 19.24(a), we first enter the expressions of the objective and constraint functions, then identify the bounds of design variables ($x1$, $x2$, $f1$, and $f2$ correspond respectively to the two design variables a and h and the two objectives A and T in the pyramid example). Note that a starting point (initial design) is required (in this case, we choose $a = 15$ and $h = 15$) to initiate the optimization. The first solution obtained is shown in Figure 19.24(b) ($A = 608.3$, $T = 883.5$), which represents the black dot on the Pareto front (Figure 19.25(a)). The points A^* (594.8, 938.2)

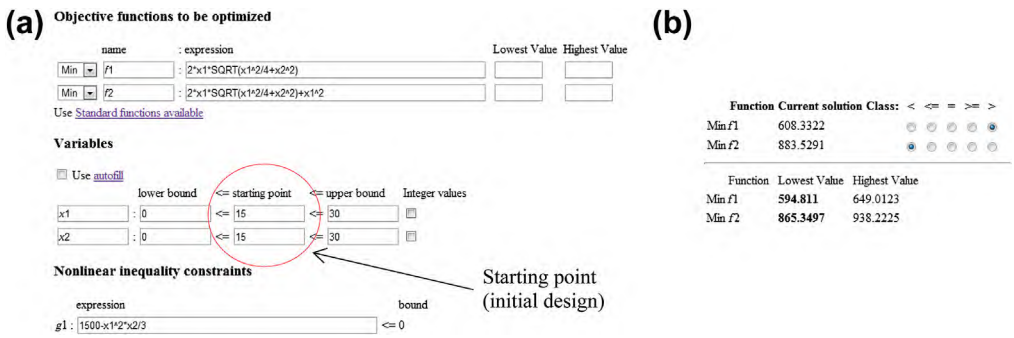


FIGURE 19.24

Screen shots of NIMBUS for the pyramid example. (a) Defining the optimization problem. (b) Interface of classification with the first Pareto solution and the minimum of individual objectives listed.

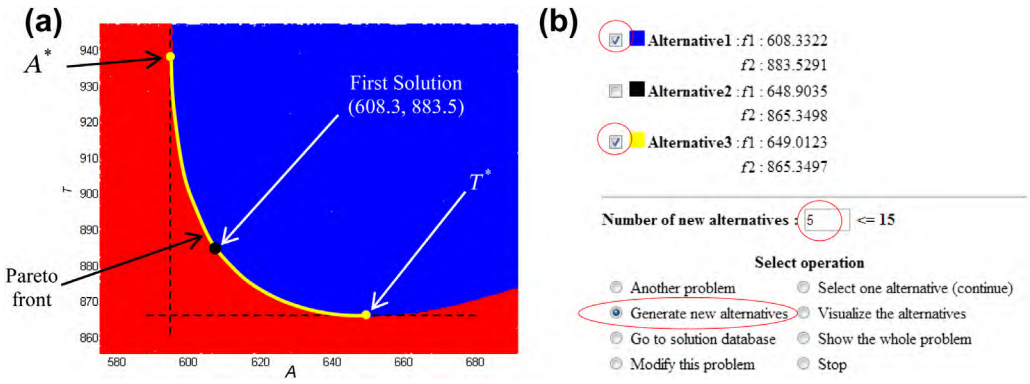


FIGURE 19.25

Sample windows of NIMBUS software. (a) Pareto front of the pyramid example with the first solution obtained. (b) New solutions generated based on the first classification iteration.

and T^* (649.0, 865.3) corresponding to the minimum of individual objectives are also calculated automatically, and are listed under the first solution, as shown in Figure 19.24(b).

The next step is classification. As can be seen in Figure 19.24(b), a group of buttons to the right of the first solution allow us to choose the "class" for each objective function. In this case, we choose "$>$" for $f1$ (objective function A) and "$<$" for $f2$ (objective function T), which implies our preference—further minimizing $f2$ based on the current solution while allowing $f1$ to change freely. Alternatively, if, for example, "$\geq$" is chosen for $f1$, then a value needs to be assigned as the maximum allowed value for $f1$ when minimizing $f2$. The two new solutions (alternatives 2 and 3) generated according to the classification information are shown in Figure 19.25(b). Note that $f2$ has been minimized to its lower bound in alternative 3. We then choose alternatives 1 and 3 to create five more Pareto solutions between the two (Figure 19.25(b)), and the results are listed in Figure 19.26(a) as alternatives 2 to 6.

A number of solution visualization methods are available in NIMBUS, two of which are demonstrated in Figures 19.26(b) and (c). In Figure 19.26(c), each line represents a solution on the Pareto front (the seven solutions so far) between the first solution (alternative 1) and point T^* (alternative 7). We can certainly continue the optimization process to obtain additional Pareto solutions.

Note that for nonacademic users, an implementation of NIMBUS that operates under several operating systems (such as Linux and Windows) is also available, known as IND-NIMBUS (ind-nimbus.it.jyu.fi). IND-NIMBUS has been tested with several industrial problems and can be connected with different simulators or modeling tools, such as MATLAB.

19.5.2 COMMERCIAL TOOLS

There are several general-purpose multiobjective optimization software packages currently available in the commercial sector. In general, commercial optimization software tools provide intuitive user interfaces with shorter learning curves than academic codes. These software programs share similar

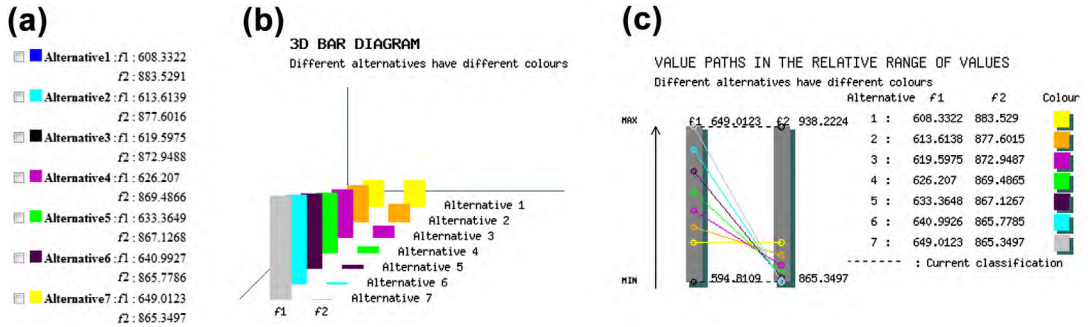


FIGURE 19.26

Multiobjective optimization result. (a) Seven Pareto solutions obtained. (b) Visualization using a three-dimensional bar diagram. (c) Visualization using value paths.

major characteristics, while some of them also offer unique capabilities. Examples of such software include the following:

- modeFRONTIER®, developed by ESTECO (www.esteco.com/modefrontier)
- OPTIMUS®, developed by Noesis Solutions (www.noesissolutions.com/Noesis/about-optimus)
- optiSLang®, developed by Dynardo (www.dynardo.de/en/software/optislang.html)
- BOSS Quattro®, developed by LMS (www.lmsintl.com/samtech-boss-quattro)
- Isight®, developed by Dassault Systèmes (www.modelon.com/products/isight).

One of the key advantages of commercial codes is their ability to seamlessly couple with multiple third-party (commercial and in-house) engineering tools. Unlike the academic codes, these commercial software packages are designed to work with a wide variety of modeling and simulation software programs. Each of them can be thought of as an integration platform that enables the automation of a complex design simulation, and in the meantime offers optimization capabilities to support decision making. For example, the workflow environment in modeFRONTIER® is able to formalize, drive, and manage individual steps, such as CAD modeling, computation fluid dynamics (CFD), finite element analysis (FEA), and so on, composing an engineering process through the integration of the most popular engineering solvers, while the data can be transferred automatically from one simulation to the next. Such communication between software is usually guaranteed by application protocol interfaces or automatic file exchange. Figure 19.27 shows a screenshot of the Workflow Editor in modeFRONTIER®, and some of the third-party software supported in modeFRONTIER®, including CAD, CAE, and generic applications, are listed in Table 19.4. Additional wizard style tools are also available in modeFRONTIER® to achieve the coupling with other commercial tools or in-house codes. Some commercial optimization software, such as OPTIMUS® and BOSS Quattro®, also provide native interfaces with major CAE and FEA systems, broadening the basic capabilities of those engineering tools.

In addition to a strong connection with third-party applications, each of the commercial software packages mentioned above also bundles a collection of advanced optimization techniques for both single- and multiobjective problems, ranging from gradient-based methods to genetic algorithms.

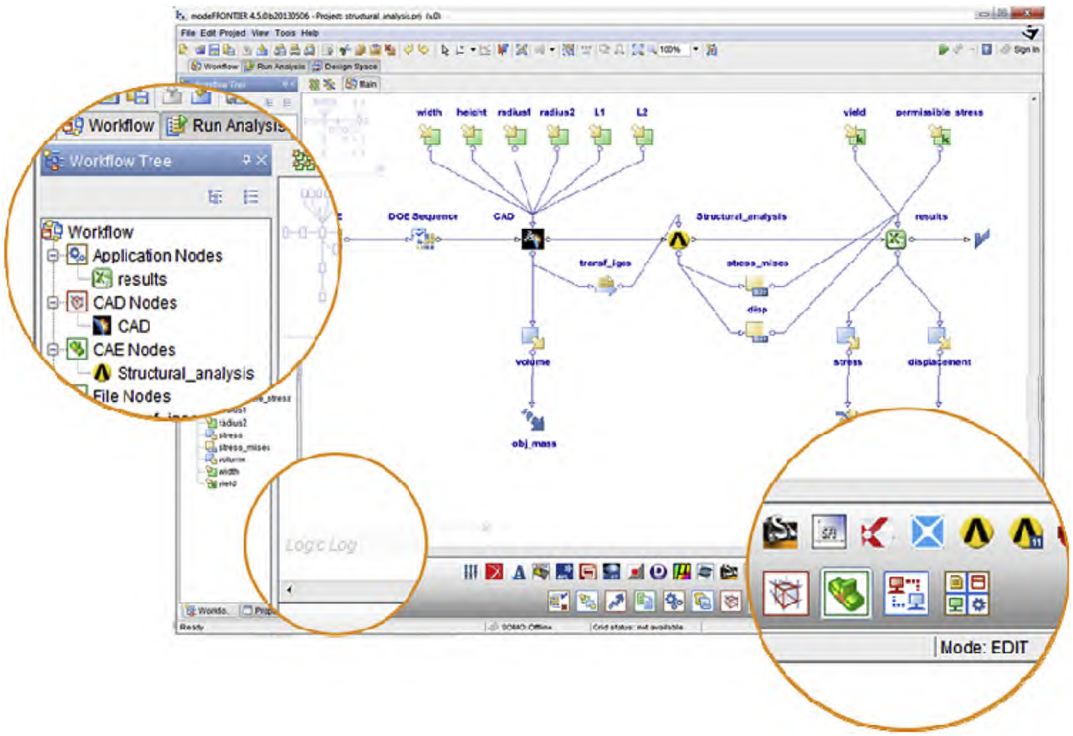


FIGURE 19.27

Workflow Editor in modeFRONTIER® (www.esteco.com/modefrontier/modefrontier-platform-overview).

Table 19.4 Examples of Third-Party Applications Supported in modeFRONTIER®	
CAD	SolidWorks, CATIA, Creo, NX, Spaceclaim
CAE	ANSYS Workbench, ABAQUS, Adams, Virtual.Lab, Image.Lab, ANSA
Generic applications	Excel, OpenOffice, SCIBAL, MATLAB, LABVIEW, MATHCAD

For example, the multiobjective algorithms available in modeFRONTIER® include multiobjective genetic algorithm (MOGA), adaptive range MOGA, multiobjective simulated annealing (MOSA), nondominated sorting genetic algorithm (NSGA II), multiobjective game theory, evolutionary strategies methodologies, and normal boundary intersection. Moreover, in modeFRONTIER®, different algorithms can even be combined by the user in order to obtain hybrid approaches. For example, it is possible to combine the robustness of a genetic algorithm together with the accuracy of a gradient-based method, using the former for initial screening and the latter for refinements, so that the efficiency of the design cycle can be further enhanced. During optimization, these commercial software

codes continuously update product parameters and extract relevant outputs from individual steps within the workflow until an optimal design is obtained. Because running a large number of simulations for optimization can be computationally expensive in practical engineering design, response surface methods (Jones, 2001; Poles et al., 2008) are employed in most commercial software packages to accelerate the optimization process.

Another advantage of commercial optimization software over academic tools is that they often provide interactive result viewers to help users select the most suitable design. The alternatives obtained during optimization can be displayed in the forms of tables, charts, or two-dimensional (2D) and three-dimensional (3D) plots. As an example, the visualization of 2D and 3D Pareto fronts in OPTIMUS® is shown in Figure 19.28. Usually, if the problem definition exceeds three dimensions, a 2D or 3D subspace can be selected, and parallel coordinate plots can be used to determine the best designs out of the set of Pareto designs. Information other than the optimal solutions, such as the design space and sensitivity, can also be extracted and analyzed using the visualization tools incorporated in individual commercial optimization software (for example, see Figure 19.29).

Note that additional capabilities other than those discussed above are available in most commercial optimization software packages. For example, in modeFRONTIER®, OPTIMUS®, optiSLang®, and Isight®, robustness analysis and reliability analysis can be performed based on statistical methods, such as the Monte Carlo method, to deal with uncertainties that impact the optimization process. Also, modeFRONTIER®, BOSS Quattro®, and Isight® allow users to run multiobjective optimization in parallel by evaluating more than one simulation at the same time using several local or remote processors.

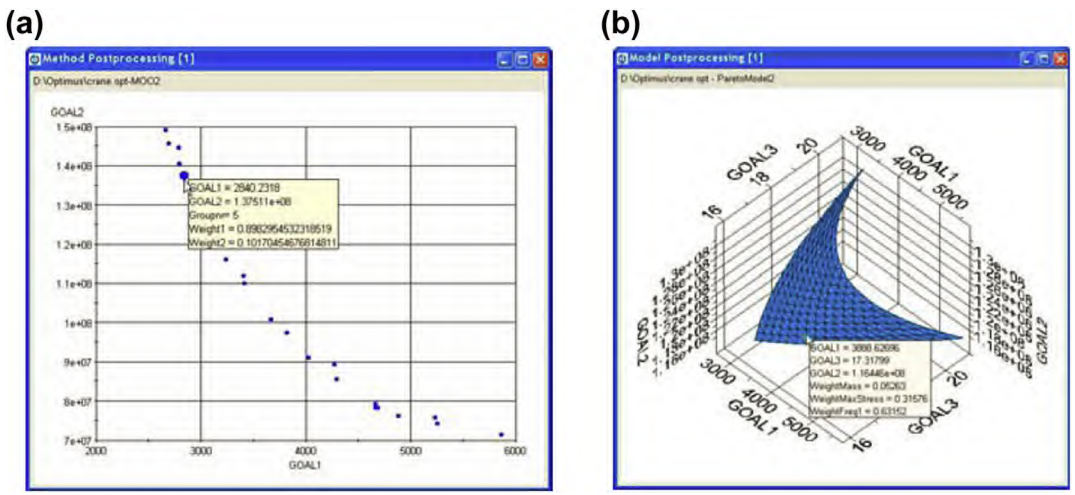


FIGURE 19.28

Optimization result visualization in OPTIMUS®. (a) A scatter plot showing points on the Pareto optimal set for two objectives. (b) A 3D plot of the Pareto optimal set (Poles et al., 2008).

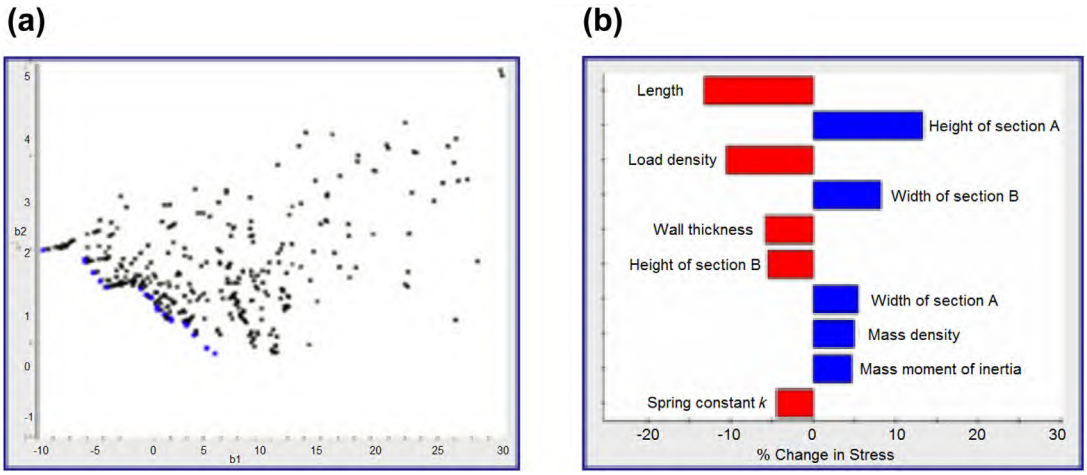


FIGURE 19.29

Optimization result visualization in Isight™. (a) A scatter plot of the design space. (b) Parameter effects on the performance measure (www.modelon.com/products/isight/).

19.6 ADVANCED TOPICS*

Before closing out this chapter, we include two advanced topics that are relevant to design optimization. The first topic is reliability-based design optimization (RBDO), which incorporates variations in physical parameters and design variables into design optimization, leading to design with less probability of failure. Second, we introduce a case of design optimization that takes product cost, including manufacturing, as an objective function. This case represents a more realistic design optimization problem that leads to less expensive designs.

19.6.1 RELIABILITY-BASED DESIGN OPTIMIZATION

In Chapter 10, we discussed reliability analysis. We introduced the concept of failure modes associated with certain critical product performance. We incorporated the variability (or uncertainty) of physical parameters or a manufacturing process that affects the performance (hence, failure modes) of the product to estimate a failure probability. We used a beam example for illustration, in which the failure probability of a stress failure mode predicts the percentage of the incidents when the maximum stress of the beam exceeds its material yield strength. The result offered by the reliability analysis is far more precise and effective than that of the safety factor approach.

In this subsection, we move one step further to discuss design optimization by incorporating failure modes (and failure probabilities) into optimization problem formulation. We first briefly review the basics of the reliability analysis, in particular, the most probable point (MPP) search for failure probability estimates. We formulate the mathematic equations for a standard RBDO and its solution technique, and we present a sample case to illustrate the topic of RBDO using a tracked-vehicle

roadarm example. Readers are encouraged to review Chapter 10 to refresh the concept and numerical computations involved in the reliability analysis before reading further.

19.6.1.1 Failure Probability

The probability of failure P_f of a product with a failure mode $g(\mathbf{X}) \leq 0$ is defined as

$$P_f = P(M \leq 0) = P(g(\mathbf{X}) \leq 0) = \int_{g(\mathbf{x}) \leq 0} \mathbf{f}_{\mathbf{X}}(\mathbf{x}) \mathrm{d}x \tag{19.35}$$

where $f_{\mathbf{X}}(\mathbf{x})$ is the joint probability density function (PDF), $\mathbf{X}$ is a vector of random variables, and the function $g(\mathbf{x}) = 0$ is called the limit state function. Note that realization of $\mathbf{X} = [X_1, X_2, \ldots, X_n]^{\mathrm{T}}$ is denoted as $\mathbf{x} = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}$, which is a point in the n-dimensional space.

The probability integration in Eq. 19.35 is visualized for a two-dimensional case in Chapter 10, Figure 10.11(a), which shows the joint PDF $f_{\mathbf{X}}(\mathbf{x})$ and its contour projected onto the x_1-x_2 plane. All the points on the projected contours have the same values of $f_{\mathbf{X}}(\mathbf{x})$ or the same probability density. The limit state function $g(\mathbf{x}) = 0$ is also shown. The failure probability P_f is the volume underneath the surface of the joint PDF $f_{\mathbf{X}}(\mathbf{x})$ in the failure region $g(\mathbf{x}) \leq 0$. To show the integration more clearly, the contours of the joint PDF $f_{\mathbf{X}}(\mathbf{x})$ and the limit state function $g(\mathbf{x}) = 0$ are plotted on the x_1-x_2 plane, as shown in Chapter 10, Figure 10.11(b).

The direct evaluation of the probability integration of Eq. 19.35 is very difficult if not impossible. A number of methods have been developed. Monte Carlo simulation is simple and easy to implement. However, this brute-force approach requires tens of thousands of analyses or more, which is impractical for engineering problems that often require significant computational effort. One of the widely employed methods to alleviate the computational issue to some extent, while still offering a sufficiently accurate estimate on the failure probability, is the first-order reliability method (FORM) or second-order reliability method (SORM).

As discussed in Chapter 10, the key idea in calculating the failure probability using FORM or SORM is to locate the MPP in the U-space. Many numerical approaches have been developed for the MPP search. These methods can be categorized in two major categories: the reliability index approach (RIA) and the performance measure approach (PMA). The reliability index approach employs a forward reliability analysis algorithm that computes failure probability for a prescribed performance level in the limit state function. The performance measure approach employs an inverse reliability analysis algorithm that computes response level for a prescribed failure probability.

Next, we formulate the standard reliability-based design optimization problems, in which we assume the RIA for the MPP search.

19.6.1.2 RBDO Problem Formulation

The classical design optimization problem based on deterministic analysis is typically formulated as a nonlinear constrained optimization problem, as discussed in Chapter 17. Similarly, the RBDO problem can also be formulated as a nonlinear constrained optimization problem where reliability measures are included as constraint functions. Probabilistic constraints in RBDO ensure a more evenly distributed failure probability in the component of a product. In general, the RBDO model contains two types of design variables: distributional design variable $\boldsymbol{\theta}$ and conventional deterministic design variable $\mathbf{b}$.

Let $\boldsymbol{\theta} = [\theta_1,\theta_2,...,\theta_{n1}]^T$ and $\mathbf{b} = [b_1,b_2,...,b_{n2}]^T$ be the distributional and deterministic design variable vectors of dimensions n_1 and n_2, respectively. The RBDO problem can be formulated as follows:

$$\text{Minimize: } f(\boldsymbol{\theta}, \mathbf{b}) \tag{19.36a}$$

$$\text{Subject to: } P_{f_i} = P\big(g_i(\boldsymbol{\theta}, \mathbf{b}) \le 0\big) - P_i^u \le 0, \quad i = 1, m \tag{19.36b}$$

$$\theta_j^\ell \le \theta_j \le \theta_j^u, \quad j = 1, n_1 \tag{19.36c}$$

$$b_k^\ell \le b_k \le b_k^u, \quad k = 1, n_2 \tag{19.36d}$$

where $f(\boldsymbol{\theta}, \mathbf{b})$ is the objective function, $P(\bullet)$ denotes the probability of the event $(\bullet)$, and P_i^u is the required upper bound of the probability of failure for the ith constraint function P_{fi}. In Eqs (19.36c) and (19.36d), θ_j^ℓ and θ_j^u, and b_k^ℓ and b_k^u, are the lower and upper bounds of the jth distributional and kth deterministic design variables, respectively.

The reliability constraints defined in Eq. 19.36b are assumed to be independent and thus no correlation exists. As mentioned, it is almost impossible to calculate the probability of failure P_{fi} in Eq. 19.36b by a multiple integration for general design applications. Consequently, the FORM or other more efficient reliability analysis methods are employed. The computational flow of RBDO using the FORM is illustrated in Figure 19.30. Note that at each design iteration, the FORM needs to be carried out several times for individual failure functions. As formulated in Chapter 10, each FORM is equivalent to a deterministic optimization, which is very computationally demanding. This is the reason why RBDO is mostly limited to academic problems. Furthermore, the first-order derivative of the failure probability with respect to both distributional and deterministic design variables must be computed to support gradient-based RBDO.

19.6.1.2.1 Reliability-Based Design Sensitivity Analysis

The sensitivity of the failure probability includes two parts: the sensitivity of the failure probability with respect to the distributional parameters $\boldsymbol{\theta}$ of random variables (e.g., mean value, standard deviation), and the sensitivity of the failure probability with respect to the deterministic design variables $\mathbf{b}$.

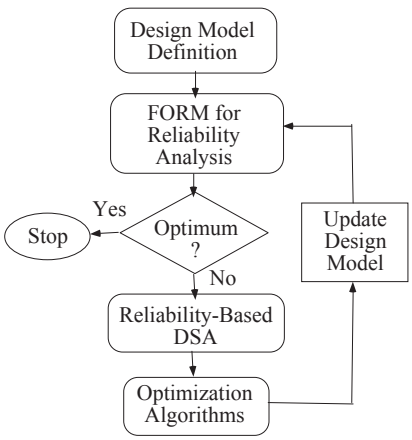


FIGURE 19.30

Computation flow for gradient-based RBDO.

The derivative of the estimated failure probability P_f obtained using the FORM with respect to a design variable η, which can be either θ_j or b_k, is

$$\frac{\partial P_f}{\partial \eta} = \frac{\partial \Phi(-\beta)}{\partial \eta} = \frac{\partial \Phi(-\beta)}{\partial \beta}\frac{\partial \beta}{\partial \eta} = -\Phi(-\beta)\frac{\partial \beta}{\partial \eta} \tag{19.37}$$

where Φ is the standard normal density function. Therefore, to compute the sensitivity of the failure probability P_f, $\partial\beta/\partial\eta$ must be computed as

$$\frac{\partial \beta}{\partial \eta} = \frac{\partial \left(\mathbf{U}^{*\mathrm{T}}\mathbf{U}^*\right)^{1/2}}{\partial \eta} = \frac{1}{\beta}\mathbf{U}^{*\mathrm{T}}\frac{\partial \mathbf{U}^*}{\partial \eta} \tag{19.38}$$

where $\mathbf{U}^*$ is the MPP found in the U-space.

The sensitivity of the reliability index with respect to a distributional design variable θ_j can be obtained by substituting $\eta = \theta_j$ and $\mathbf{U}^* = \mathbf{T}(\mathbf{X}^*\boldsymbol{\theta})$ in Eq. 19.38 as

$$\frac{\partial \beta}{\partial \theta_j} = \frac{1}{\beta}\mathbf{U}^{*\mathrm{T}}\left(\frac{\partial \mathbf{T}(\mathbf{X}^*, \boldsymbol{\theta})}{\partial \theta_j} + \frac{\partial \mathbf{T}(\mathbf{X}^*, \boldsymbol{\theta})}{\partial \mathbf{X}^*}\frac{\partial \mathbf{X}^*}{\partial \theta_j}\right) = \frac{1}{\beta}\mathbf{U}^{*\mathrm{T}}\frac{\partial \mathbf{T}(\mathbf{X}^*, \boldsymbol{\theta})}{\partial \theta_j} \tag{19.39}$$

in which the second term in the parenthesis vanishes, as proven in Yu (1996). For the normally distributed random variables, where $\mathbf{T}$ can be explicitly written as a transformation function of $\boldsymbol{\theta}$, Eq. 19.39 can be calculated analytically. For the non-normally distributed random variables, the transformation $\mathbf{T}$ cannot be obtained explicitly. In such a case, the finite difference method can be used to approximate the derivative of $\mathbf{T}$ with respect to θ_j.

As discussed, the reliability index β is the distance between the origin and the MPP in the U-space. The MPP vector $\mathbf{U}^*$ on the failure surface can be written as

$$\mathbf{U}^* = -\beta\frac{\nabla g(\mathbf{U}^*, \mathbf{b})}{|\nabla g(\mathbf{U}^*, \mathbf{b})|} \tag{19.40}$$

in which, $\nabla g(\mathbf{U}^*,\mathbf{b})$ is the gradient of the failure function at the MPP:

$$\nabla g(\mathbf{U}^*, \mathbf{b}) = \frac{\partial g(\mathbf{U}^*, \mathbf{b})}{\partial \mathbf{U}} \tag{19.41}$$

From Eq. 19.40, the MPP vector $\mathbf{U}^*$ is also a function of $\mathbf{b}$ because the failure function g depends on the deterministic design variables $\mathbf{b}$. Substituting Eq. 19.40 into Eq. 19.38 yields

$$\frac{\partial \beta}{\partial \mathbf{b}} = -\frac{1}{|\nabla g(\mathbf{U}^*, \mathbf{b})|}\frac{\partial g^{\mathrm{T}}(\mathbf{U}^*, \mathbf{b})}{\partial \mathbf{U}}\frac{\partial \mathbf{U}^*}{\partial \mathbf{b}} \tag{19.42}$$

By taking the derivative of $g(\mathbf{U}^*, \mathbf{b}) = 0$ with respect to $\mathbf{b}$,

$$\frac{\partial g^{\mathrm{T}}(\mathbf{U}^*, \mathbf{b})}{\partial \mathbf{U}}\frac{\partial \mathbf{U}^*}{\partial \mathbf{b}} + \frac{\partial g(\mathbf{U}^*, \mathbf{b})}{\partial \mathbf{b}} = 0 \tag{19.43}$$

Substituting Eq. 19.43 into Eq. 19.42 yields

$$\frac{\partial \beta}{\partial \mathbf{b}} = \frac{1}{|\nabla g(\mathbf{U}^*, \mathbf{b})|}\frac{\partial g^{\mathrm{T}}(\mathbf{U}^*, \mathbf{b})}{\partial \mathbf{b}} \tag{19.44}$$

Note that evaluation of Eq. 19.44 needs only the first-order derivative of the failure function with respect to deterministic design variables.

To avoid the prohibitively expensive computational efforts required for a large number of reliability analyses during a batch-mode RBDO for design optimization, a mixed-design approach (Yu et al., 1997) that includes deterministic design optimization in batch mode and reliability-based design in an integrated mode, as discussed in Chapter 17, is employed. The mixed design approach starts with a deterministic design optimization, in which performance measures employed as failure modes are defined as constraint functions. After an optimal design is obtained, a reliability analysis is performed to ascertain if the deterministic optimal design is reliable. If the probability of the failure of the deterministic optimal design is found to be unacceptable, a reliability-based design approach that employs a set of interactive design steps, such as trade-off analysis and what-if study, is used to obtain a near-optimal design that is reliable with an affordable computational cost. A tracked-vehicle roadarm is employed to illustrate the approach next.

19.6.1.3 RBDO for a Tracked-Vehicle Roadarm

A roadarm of the military tracked-vehicle shown in Figure 9.26 is employed to illustrate the mixed design approach for the RBDO. A deterministic design optimization is presented first. Then, reliability analysis using the FORM is discussed. The reliability-based design obtained using the interactive design process follows.

19.6.1.3.1 Deterministic Design Optimization

A 17-body dynamic simulation model discussed in Section 9.8.1 (Figure 9.26) is created to drive the tracked vehicle on a proving ground, at a constant speed of 20 miles per hour. A 20-s dynamic simulation is performed at a maximum integration time step of 0.05 sec. using dynamic simulation and design system or DADS (Haug and Smith, 1990). The joint reaction forces applied at the wheel end of the roadarm, accelerations, angular velocities, and angular accelerations of the roadarm are obtained from the dynamic simulation. Four beam elements, STIF4, and 310 20-node isoparametric finite elements, STIF95, of ANSYS are used for the roadarm finite element model shown in Chapter 10, Figure 10.24(a). The roadarm is made of S4340 steel and the length between the centers of the two holes is 20 in.

The fatigue life fringe plot is shown in Figure 19.31. At the initial design, the structural volume is 486.7 in^3. The crack initiation lives at 24 critical points (with node IDs shown in Figure 19.31) are defined as the constraints with a lower bound of 9.63 $\times$ 106 blocks (20 sec. per block). Note that the lower bound defined is equivalent to 20 years service life, assuming the tracked vehicle is operated 8 hrs. per day, 5 days per week. Definitions of the objective function and five critical constraint functions are listed in Table 19.5.

For shape design parameterization, eight design variables are defined to characterize the geometric shapes of the four sections, as shown in Chapter 10, Figure 10.24(b). The profile of the crosssection shape is composed of four straight lines and four cubic curves. Side expansions (x_1' – direction) of cross-sectional shapes are defined using design variables b1, b3, b5, and b7 for intersections 1 to 4, respectively. Vertical expansions (x_3' – direction) of the cross-sectional shapes are defined using the remaining four design variables.

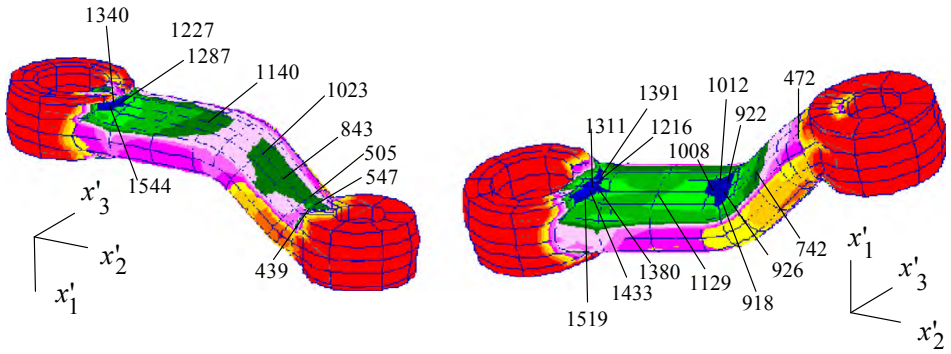


FIGURE 19.31

Fringe plots of crack initiation life.

Table 19.5 Objective and Critical Constraint Functions

Function	Description	Lower Bound	Current Design	Status
Objective	Volume		487.678 $in.^3$	
Constraint 1	Node 1216	9.63×10^6 (20 years)	9.631×10^6 bks	Active
Constraint 2	Node 926	9.63×10^6 (20 years)	8.309×10^7 bks	Inactive
Constraint 3	Node 1544	9.63×10^6 (20 years)	8.926×10^7 bks	Inactive
Constraint 4	Node 1519	9.63×10^6 (20 years)	1.447×10^8 bks	Inactive
Constraint 5	Node 1433	9.63×10^6 (20 years)	2.762×10^8 bks	Inactive

A deterministic optimal design is obtained in six design iterations using the modified feasible direction method in the design optimization tool (DOT). As shown in Table 19.6, at the deterministic optimal design, all fatigue lives are greater than the lower bound and objective function is reduced by 10.5%. The geometric shapes of the roadarm at initial and deterministic optimal designs are shown in Figure 19.32.

19.6.1.3.2 Probabilistic Fatigue Life Predictions

The random variables and their statistical values for the crack initiation life predictions are listed in Chapter 10, Table 10.5, including material and tolerance random variables. The eight tolerance random variables b1 to b8 are defined corresponding to the eight shape design variables defined in Chapter 10, Figure 10.24(b).

FORM is used to calculate the reliability of the crack initiation life at five critical points. The results shown in Table 19.7 indicate that the failure probability at nodes 926 and 1544 is greater than 3%. Because the failure probability of the roadarm at the deterministic optimal design is too high, a reliability-based design must be conducted to reduce the failure probability (to obtain a feasible design).

Table 19.6 Objective and Critical Constraint Function Values at Initial and Deterministic Optimal Designs

Function	Description	Initial Design	Deterministic Optimal Design	Changes
Objective	Volume	487.678 $in.^3$	436.722 $in.^3$	−10.5%
Constraint 1	Node 1216	9.631×10^6 bks	7.704×10^7 bks	699.9%
Constraint 2	Node 926	8.309×10^7 bks	9.631×10^6 bks	−88.4%
Constraint 3	Node 1544	8.926×10^7 bks	9.678×10^6 bks	−89.2%
Constraint 4	Node 1519	1.447×10^8 bks	4.698×10^7 bks	−67.5%
Constraint 5	Node 1433	2.762×10^8 bks	4.815×10^8 bks	74.3%

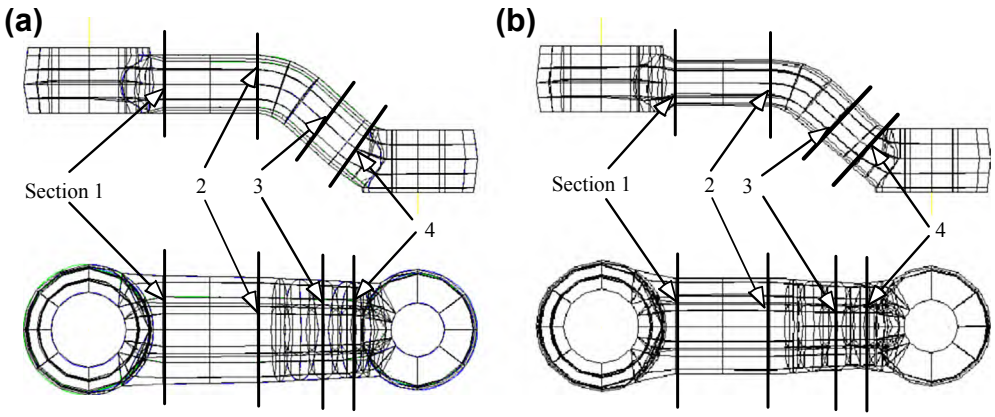


FIGURE 19.32

Geometric shape of the roadarm in front and top views. (a) Initial design. (b) Deterministic optimal design.

Table 19.7 Objective and Failure Function Values at Deterministic Optimal and Improved Designs

Function	Description	Deterministic Optimal Design	Improved Design (2 RBDOs)	Changes
Objective	Volume	436.722 $in.^3$	447.691 $in.^3$	2.5%
Constraint 1	Node 1216	0.476%	0.532%	
Constraint 2	Node 926	3.24%	0.992%	
Constraint 3	Node 1544	3.21%	0.998%	
Constraint 4	Node 1519	0.83%	0.721%	
Constraint 5	Node 1433	0.023%	0.018%	

Table 19.8 Design Variable Values at Initial, Deterministic Optimal, and Improved Designs

Design Variables	Initial Design	Deterministic Optimal Design	Improved Design
b1 (in.)	3.250	2.889	2.902
b2 (in.)	1.968	1.583	1.593
b3 (in.)	3.170	2.911	2.925
b4 (in.)	1.968	1.637	1.687
b5 (in.)	3.170	2.870	2.904
b6 (in.)	2.635	2.420	2.442
b7 (in.)	3.170	2.801	2.881
b8 (in.)	5.057	4.700	4.700

19.6.1.3.3 Reliability-Based Design

For the reliability-based design, the mean values of the eight shape parameters shown in Figure 10.24(b) are chosen as the design variables. The objective function is still the structural volume. The constraint functions are the failure probability of the fatigue life at the five critical points, with an upper bound of 1% (i.e., the required reliability of fatigue life larger than 20 years is 99%). Table 19.7 shows that the initial design is infeasible because the second and third constraints are violated. The reliability-based design sensitivity analysis (DSA) method discussed above is used to calculate the sensitivity coefficients of the fatigue failure probability with respect to the design variables.

Because the current design is infeasible, a constraint correction algorithm is selected for the trade-off analysis. Using the sensitivity coefficients, a QP (quadratic programming) subproblem is employed to search a direction in which the reliability will quickly increase. Then, a what-if study is performed along the search direction suggested by the trade-off study, plus a step size.

Through two iterations, a feasible design is achieved, as shown in Table 19.8. The two design iterations took 10 FORMs and two reliability-based DSAs. At the improved design, failure probabilities at five critical points are less than 1%, with 2.5% increments in volume. However, the total volume savings starting from the initial design is 8%—that is, from 487 in^3 to 447 in^3. The design variable values of the initial, deterministic optimal, and improved (after two interactive RBDOs) designs are listed in Table 19.8.

19.6.2 DESIGN OPTIMIZATION FOR STRUCTURAL PERFORMANCE AND MANUFACTURING COST

In mechanical and aerospace industries, engineers often confront the challenge of designing components (e.g., automotive suspension and engine components) that can sustain structural loads and meet the functional requirements. It is imperative that these components contain the minimum material to reduce cost and increase efficiency of the mechanical system, such as fuel consumption. The geometry of these components is usually complicated due to strength and efficiency requirements, which often results in increased manufacturing time and cost. Some of these structural components are shown in Figure 19.33.

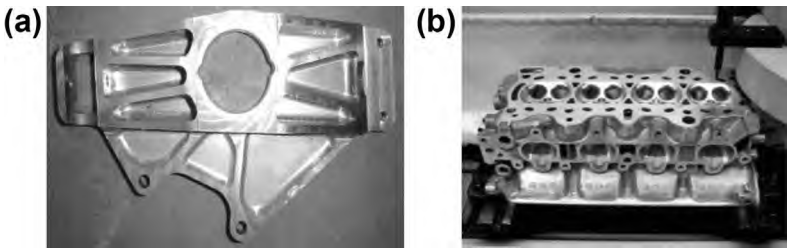


FIGURE 19.33

Mechanical components involving time-consuming and precision machining operations. (a) Upright in an automotive suspension. (b) Engine block.

Although structural optimization has been widely used for many decades to solve such problems, the primary focus has been on the functionality aspects of the design. During the course of an optimization, the geometric complexity of the component may increase, making manufacturing difficult or uneconomical. Due to the increasing geometric complexity, conventional structural optimization problems that define mass as the objective function may not yield components with minimum cost from the manufacturing perspective. This especially holds true for machined components because machining cost is often the dominant constituent of product cost for such components.

In this subsection, we present a case study of structural shape optimization that incorporates machining and material costs as the objective function subject to structural performance constraints. Structural shape optimization reduces material, but it may be accompanied by increases in the geometric complexity due to changes in the design boundary, which ultimately increases manufacturing cost. In this case study, we present a design process that incorporates manufacturing costs into structural shape optimization and produces components that are cost effective and satisfy specified structural performance requirements.

19.6.2.1 Design Problem Definition and Optimization Process

As discussed in Chapter 17, a typical single-objective optimization problem is defined as follows:

$$\text{Minimize: } f(\mathbf{b}) \tag{19.45a}$$

$$\text{Subject to: } g_i(\mathbf{b}) \leq 0, \quad i = 1, m \tag{19.45b}$$

$$h_j(\mathbf{b}) = 0, \quad j = 1, p \tag{19.45c}$$

$$b_k^{\ell} \leq b_k \leq b_k^{u}, \quad k = 1, n \tag{19.45d}$$

where $f(\mathbf{b})$ is the objective function; $\mathbf{b}$ is the vector of design variables, $g_i(\mathbf{b})$ is the ith inequality constraint, and $h_j(\mathbf{b})$ is the jth equality constraint. The objective function $f(\mathbf{b})$ is the product cost for the component to be discussed shortly.

It is assumed that the designer has all of the required data, such as the initial shape of the component, boundary and loading conditions, material properties, and machining sequences. A solid model of the component is created using solid features in CAD software. Dimensions of the solid features also serve as design variables for the optimization problem. A virtual machining (VM) model

is created by defining appropriate machining operations and sequences based on the given solid model. The machining parameters specific to each machining sequence are also specified in the VM model. Similarly, an FEA model is constructed using given boundary and loading conditions, as well as initial geometric information.

The design velocity field is then computed for sensitivity analysis and finite element mesh updates. FEA and VM are conducted to evaluate structural and machining performance measures, respectively. Machining time obtained from the VM model is important for the calculation of machining costs. Data obtained from FEA and VM models are used to evaluate objective and constraint functions. Design sensitivity analysis is conducted to compute the gradients of the objective function and constraints with respect to changes in the design variables. The gradients and values of the objective and constraint functions are passed to an optimization algorithm, which determines the design changes for the next design iteration. FEA and VM models are then updated using these changes, and the process is iterated as shown in Figure 19.34, until an optimal design is achieved.

19.6.2.2 Manufacturing Cost Model

The objective function $f(\mathbf{b})$ in Eq. 19.45a is defined as follows (Edke and Chang, 2006):

$$f(\mathbf{b}) = C_{\text{mat}}\gamma V(\mathbf{b}) + C_{\text{mc}}\tau(\mathbf{b}) + C_t \tag{19.46}$$

where the three terms on the right-hand side represent material cost, machining cost, and tooling cost, respectively. In Eq. 19.46, C_{mat} is the material cost rate (\$/lb); γ is the specific weight of the material;

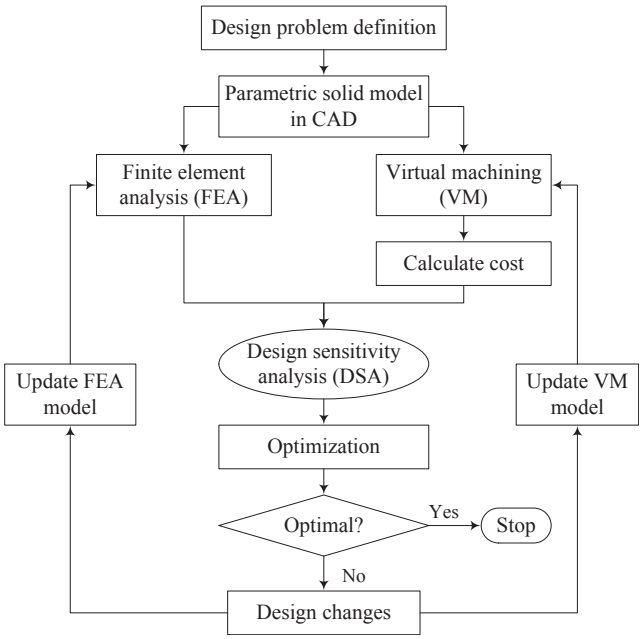


FIGURE 19.34

The design optimization process.

$V(\mathbf{b})$ is the volume of the component that depends on design; C_{mc} is the machining cost rate ($/min); $\tau(\mathbf{b})$ is the time required to machine one component, which also depends on design; and C_t is the tooling cost ($).

The machining cost rate accounts for the cost of actual machining operations, machine shop overheads, operator wages and overheads, indirect costs such as cost of electricity, and machine depreciation (Dieter, 1991). C_{mc} is given as

$$C_{mc} = \frac{1}{60}\left[\frac{M(100 + OH_M)}{100} + \frac{W(100 + OH_{OP})}{100}\right] \tag{19.47}$$

In this equation, M is the cost of machine operation per hour ($/h); OH_M is the machine overhead rate (%); W is the hourly wage for the operator ($/h); and OH_{OP} is operator overhead rate (%). The unit time $\tau(\mathbf{b})$ in Eq. 19.46 is the sum of machining time t_{mc} and idle time t_i: $\tau = t_{mc} + t_i$, in which

$$t_{mc} = \left(\frac{L}{v_f} + \frac{L'}{v_r}\right); \quad \text{and} \quad t_i = t_{set} + t_{ch} + t_{hand} + t_{down} + t_{ins} \tag{19.48}$$

In Eq. 19.48, L is the total tool travel while cutting (m); L' is the tool travel during rapid traverse (m); v_f is the feed rate (m/min); v_r is the rapid traverse velocity (m/min); t_{set} is the job setup time per part (min); t_{ch} is the time for tool change (min); t_{hand} is the work-piece handling time (min); t_{down} is the machine down time (min); and t_{ins} is the time for in-process inspection (min). The cost of tooling, which includes cost of cutting tools and cost of jigs/fixtures, is given as

$$C_t = \frac{t_{mc}}{T_{tl}}\left(\frac{K_i}{n_i} + \frac{K_h}{n_h}\right) + C_f \tag{19.49}$$

where T_{tl} is the total tool life (min); K_i is cost of the insert ($); n_i is the number of cutting edges; K_h is the tool holder cost ($); n_h is the number of cutting edges in the tool-holder; and C_f is the cost of jigs/fixtures ($). Machining time t_{mc} is obtained from VM. Similarly, cost models for other machining processes can be developed, as discussed in Chapter 15.

19.6.2.3 Virtual Manufacturing

Virtual manufacturing, as discussed in Chapter 11, is a simulation-based method that supports engineers to define, simulate, and visualize the manufacturing process in a computer environment. By using virtual manufacturing, the manufacturing process can be defined and verified early in the design process. In addition, the manufacturing time can be estimated. Material cost and manufacturing time constitute a significant portion of the product cost. The virtual machining operations, such as milling, turning, and drilling, allow designers to conduct machining process planning, generate machining tool paths, visualize and simulate machining operations, and estimate machining time. Moreover, the tool path generated can be converted into Computer Numerical Control (CNC) codes (M-codes and G-codes) (Chapter 11) and loaded to CNC machines, such as HAAS mills (www.haascnc.com), to machine functional parts as well as dies or molds for production.

Geometrically complex parts are commonly found in the automotive and aerospace industries, where molds and dies are manufactured. Typical milling operations that make molds or dies include pocket milling and surface contour milling. The quality and accuracy of the machined surfaces are

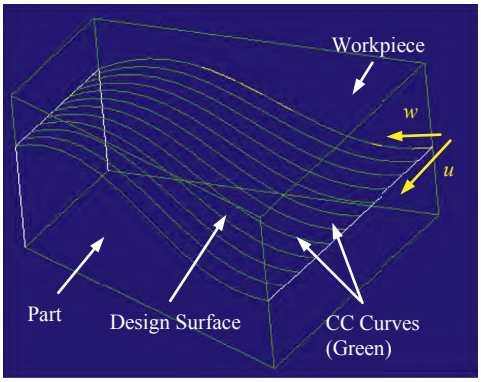


FIGURE 19.35

Design surface and CC curves.

largely determined by surface contour milling. In this section, surface contour milling will be briefly discussed to illustrate the connection between manufacturing time and structural geometric shape determined by performance requirements.

As discussed in Chapter 18, parametric surfaces are employed to parameterize the design boundary. The same surfaces will be assumed for machining. In general, the cutter contact (CC) curves are first generated on the design surface, as shown in Figure 19.35. As discussed in Chapter 12, the CC curves are created, for example, by uniformly splitting parameter u or w:

$$\mathbf{C}^i(w) = \mathbf{S}(u_i, w), \quad u_i \in [u_{\min}, u_{\max}] \tag{19.50}$$

where $\mathbf{C}^i(w)$ is the ith CC curve (which is indeed a u-isoline), and $\mathbf{S}(u, w)$ is the parametric design surface. The CC points are discrete points generated along the CC curve, which form a piecewise linear approximation of the CC curve for the CNC controller to trace. The chord length between the CC curve and the polyline formed by the CC points must be less than a prescribed tolerance, as illustrated in Figure 12.34(a). CL (cutter location) points are then obtained by offsetting from the CC points, considering workcell and cutter shape, as illustrated in Figure 19.36. Machining time t_{mc} can be estimated by the length of the CL polyline and a prescribed feedrate.

19.6.2.4 Design Sensitivity Analysis

As discussed in Chapter 18, design sensitivity analysis calculates the gradients of the objective and constraint functions with respect to design variables. Shape sensitivity analysis for structural performance measures has been developed for many years (Choi and Kim, 2006) and was briefly discussed in Chapter 18. Sensitivity of machining time due to changes in the design surface can be calculated for specific machining sequences. For example, sensitivity analysis of machining time for a contour surface milling using the isoparametric method can be obtained as follows. First, a new parametric surface will be generated for a small design change of the kth design variable δb_k. The CC curves on the perturbed parametric surface can be created by

$$\mathbf{C}^i(w; \mathbf{b} + \delta b_k) = \mathbf{S}(u_i, w; \mathbf{b} + \delta b_k), \quad u_i \in [u_{\min}, u_{\max}] \tag{19.51}$$

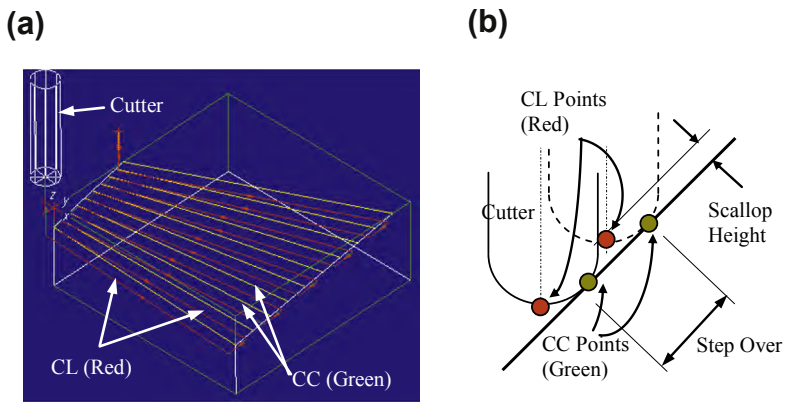


FIGURE 19.36

Offset CC for CL points: (a) CL and CC lines (3-axis mill) and (b) cutter offset and scallop.

where u_i is the u-parametric coordinate of the ith CC curve, which is kept the same before and after the design perturbation. Following the procedures discussed above, CC points and CL points are calculated next, considering chord length tolerance, workcell, and cutter shape. The new machining time $t_{\text{mc}}(\mathbf{b} + \delta b_k)$ can then be calculated by multiplying the length of the polyline formed by the new CL points with the same prescribed feedrate. The sensitivity of the machining time can be approximated by

$$\frac{\partial t_{\text{mc}}}{\partial b_k} \approx \frac{t_{\text{mc}}(\mathbf{b} + \delta b_k) - t_{mc}(\mathbf{b})}{\delta b_k} \tag{19.52}$$

However, the overall finite difference method is probably more general and more straightforward to implement in CAM, especially while considering other machining sequences, such as constant scallop height surface contour milling.

19.6.2.5 Software Implementation

The design optimization process is implemented using commercial CAD/CAM/FEA and design optimization tools, as illustrated in Figure 19.37. The shaded boxes show the commercial tools, while the plain boxes show software modules that need to be developed.

As a sample implementation (Edke and Chang, 2006), SolidWorks and Pro/ENGINEER were selected for CAD modeling, ANSYS (www.ansys.com) and Pro/MECHANICA (www.ptc.com) for finite element modeling, Pro/MFG for virtual machining, and DOT for optimization. MATLAB and C/C++ are used to construct application programs for data transfer and mathematical computations.

Note that after a change is made in model dimensions, Pro/MFG updates the toolpath for an NC sequence only after the particular NC (numerical control) sequence is run. This is when the tool path computations for that NC sequence are performed. Although most of the automation in Pro/ENGINEER can be achieved using Pro/TOOLKIT (e.g., for CAD model updates), Pro/TOOLKIT does not provide any function to perform the toolpath computations. Toolpath generation in Pro/MFG has to be carried out interactively by making a series Pro/ENGINEER menu and dialogue box selections. To overcome this problem, the "mapkeys" feature in Pro/ENGINEER is used.

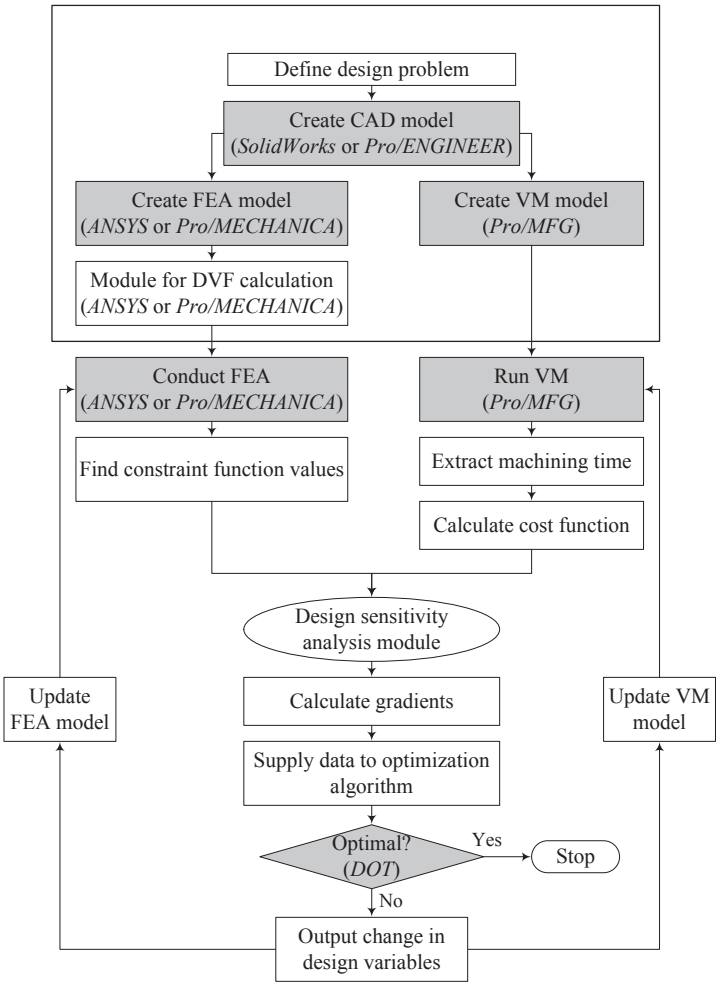


FIGURE 19.37

Optimization flow with required software modules.

Mapkeys are similar to the macros used in many application packages. A mapkey is a keyboard macro that maps frequently used command sequences to certain sets of keyboard keys. The approach for the recording of a mapkey and its value is shown in Figure 19.38 as an example. Once a mapkey is recorded, it is saved in a configuration file mapkey, with each macro beginning on a new line. Value of this mapkey (the string of commands) can be copied into a Pro/TOOLKIT application as a command string. Using a Pro/TOOLKIT function, the commands are loaded into a stack and are executed sequentially after the control returns to Pro/ENGINEER from the Pro/TOOLKIT application. Using these mapkeys, a Pro/TOOLKIT application is constructed to run the machining sequences and to extract total machining time.

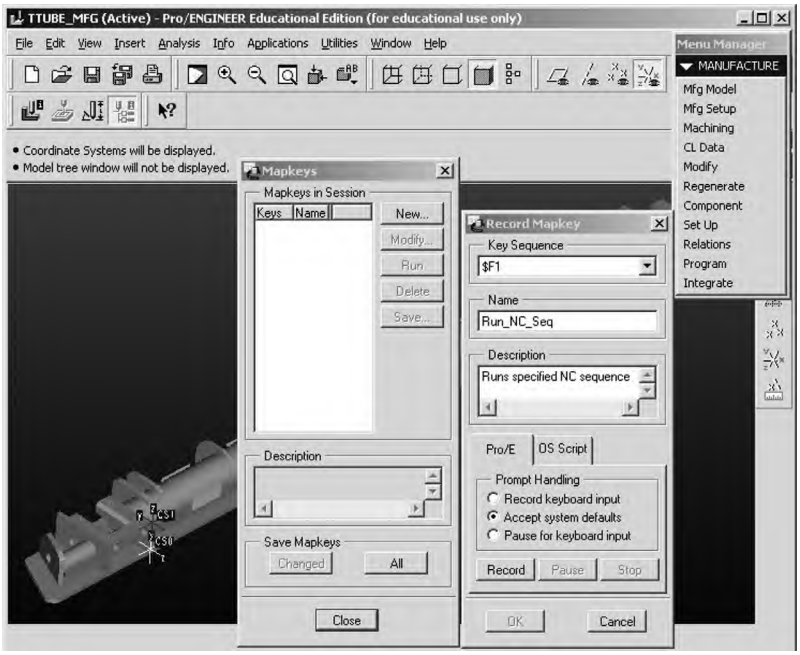


FIGURE 19.38

Recording a mapkey in Pro/ENGINEER.

19.6.2.6 Aircraft Torque Tube Example

The torque tube shown in Chapter 16, Figure 16.1 is a structural component located inside the wings of an aircraft. Loads are applied to the three brackets and the bottom face of the tube is bolted to the wing flap. The tube is made up of AL2024-T351 with yield strength of 43 ksi. Seven rectangular holes are created between fins to reduce the weight of the torque tube, as shown in Figure 19.39.

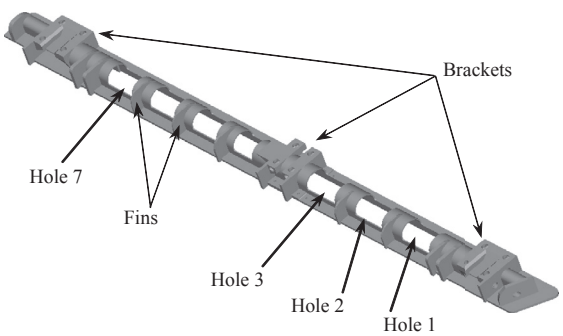


FIGURE 19.39

Torque tube with holes.

19.6.2.6.1 Problem Definition

The goal of the torque tube optimization problem is to minimize the product cost subject to limits on structural performance measures. The objective function, which is similar to Eq. 19.46, is defined as

$$\text{Minimize: } f(\mathbf{b}) = C_{\text{mat}}\gamma\{V_0 - [5V_1(b_1, b_2) - V_2(b_3, b_4) - V_3(b_5, b_6)]\} + \frac{1}{60}(t_{\text{mc}} + t_i)\left[\frac{M(100 + \text{OH}_{\text{M}})}{100} + \frac{M(100 + \text{OH}_{\text{OP}})}{100}\right] \tag{19.53a}$$

$$\text{Subject to: } \sigma_{1\max}(\mathbf{b}), \sigma_{2\max}(\mathbf{b}), \ldots, \sigma_{12\max}(\mathbf{b}) \leq 21.5 \text{ ksi} \tag{19.53b}$$

$$b_j^{\ell} \leq b_j \leq b_j^{u}, \quad j = 1, 6 \tag{19.53c}$$

Note that tooling cost is ignored in this problem for simplicity. The tube volume is computed by subtracting the volume of holes from its total volume. Since the five holes (1, 2, 5, 6, and 7) are grouped together, they have only two design variables in common (that is, b_1 and b_2). The maximum principal stresses at 12 locations are defined as constraint functions (see Figure 19.41 for some of the high stresses). The limit on the maximum stress is 21.5 ksi. It is apparent that changes in the hole sizes will vary the weight of the tube, may impact the structural integrity, and influence machining time of the tube. The design variable bounds are listed in Table 19.9.

19.6.2.6.2 Design Parameterization

Parameterization of the torque tube holes is shown in Figure 19.40. Hole depth and half of the hole length are selected as design variables. When the length of the holes is changed, the holes either expand or contract symmetrically. Also, the position of the hole is maintained such that it always remains centered between the adjacent fins. From initial tests, it was observed that the maximum stress occurs near the middle bracket. Hence, except for the two holes adjacent to the middle bracket, the design variables of all other holes are grouped together; implying that width and length of the Holes 1, 2, 5, 6, and 7 are changed at the same amounts, respectively. This reduces the number of design variables from 14 (2 design variables per hole times 7 holes) to 6.

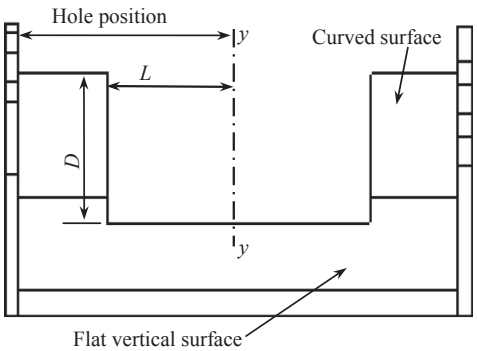


FIGURE 19.40

Parameterization of torque tube holes.

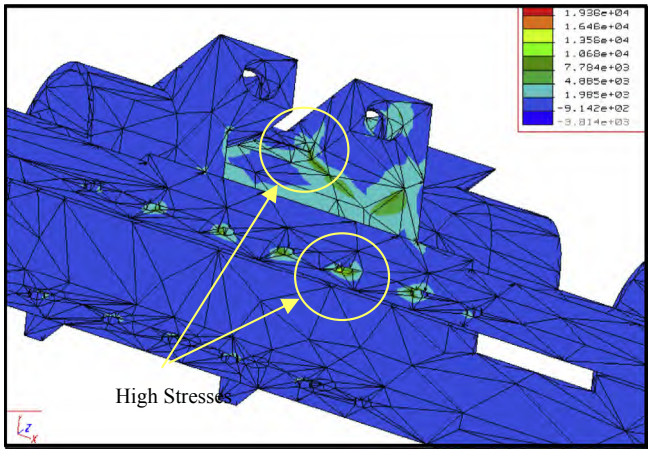


FIGURE 19.41

Finite element analysis results.

19.6.2.6.3 Finite Element Analysis

The finite element model constructed in Pro/MECHANICA consists of 11,502 elements. A p-convergence study is conducted at the outset to fix the polynomial level of the element shape function for the analysis. The criterion specified is 3.5% strain energy convergence. The polynomial order is fixed at 7. The FEA solves 2,242,623 equations. Because the model is constrained by fixing displacements at finite element nodes at some locations, concentrated high stresses at such nodes are neglected.

The highest maximum principal stress is located at one of the holes and the inner edge of the middle bracket, as shown in Figure 19.41. The maximum stress magnitude is 20.70 ksi, which is close to the constraint limit (21.5 ksi). Note that a safety factor of 2 is used.

The most critical information required for performing design optimization in this example is the design velocity field. Once the velocity field is obtained, the remaining optimization process becomes routine. For the torque tube, the design boundaries (hole surfaces) are plane surfaces, as shown in Figure 19.42. This simplifies the velocity field calculations, as the prescribed displacement itself is now the boundary velocity. Thus, only the domain velocity field calculation is required.

Table 19.9 Upper and Lower Bounds of Design Variables for the Torque Tube

	Lower Limit (in.)	Upper Limit (in.)
b1 (length of holes 1, 2, 5, 6, and 7)	0.90	2.30
b2 (depth of holes 1, 2, 5, 6, and 7)	1.50	1.95
b3 (length of hole 3)	1.10	2.50
b4 (depth of hole 3)	1.50	1.95
b5 (length of hole 4)	0.60	1.40
b6 (depth of hole 4)	1.50	1.95

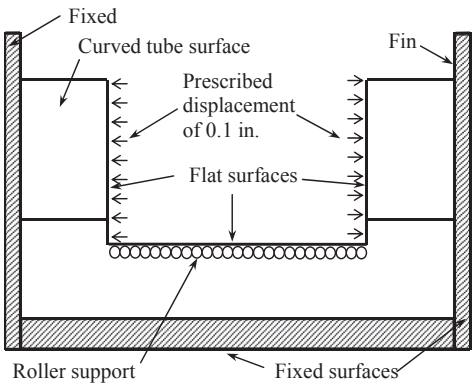


FIGURE 19.42

Velocity field computation for length design variables.

As shown in Figure 19.42, the fins and bottom surfaces are fixed. For the length design variables, a prescribed displacement of 0.1 in. is applied on the two end surfaces of the hole along the longitudinal direction. Roller boundary conditions are applied to the bottom surface of the holes. FEA is conducted to calculate the displacement of the finite element nodes, which is the design velocity field of the length design variable. To calculate the design velocity of the depth design variable, a prescribed displacement of 0.1 in. is applied to the bottom surface of the holes, and the two longitudinal end surfaces are fixed.

19.6.2.6.4 Virtual machining

The VM model defined in Pro/MFG consists of an assembly of the torque tube without holes and the torque tube with holes. The machining parameters are summarized in Table 19.10. A customized pocket milling sequence (Figure 19.43) is defined in Pro/MFG to simulate the machining process. The time required to machine all the holes is 41.26 min for the initial values of design variables.

19.6.2.6.5 Design Optimization

The sequential quadratic programming (SQP) algorithm is used for conducting design optimization. The convergence criterion is 1% of the objective function. The algorithm converges in four iterations.

Table 19.10 Machining Parameters and Machining Time

Parameter Name	Value
Tool	0.5 in. end mill
Feedrate	10 in./min
Spindle speed	1250 rpm
Step depth	0.2 in.
Machining time (initial design)	41.26 min

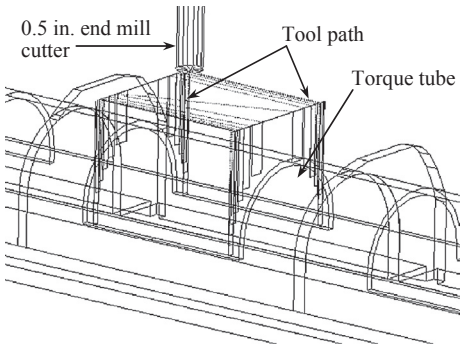


FIGURE 19.43

Virtual machining in Pro/MFG.

There is a 2.4% decrease in the cost. The weight of the torque tube reduces by 6.1%. Machining time decreases by 10.6%. The optimization history is shown in Figure 19.44. The values of the design variables for initial and final design are summarized in Table 19.11.

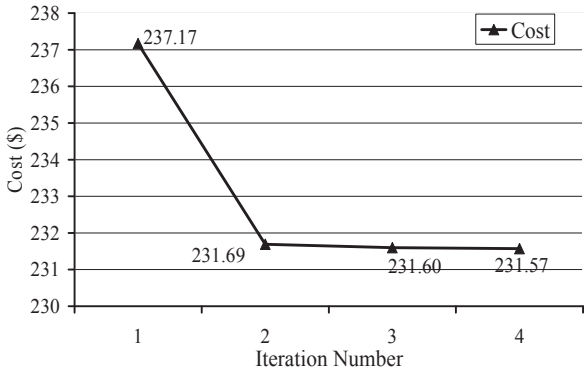


FIGURE 19.44

Optimization results for the torque tube: objective (cost $) function history.

Table 19.11 Optimization Results for the Torque Tube

	Initial Design (in.)	Final Design (in.)	% Change
b1 (length of holes 1, 2, 5, 6, and 7)	1.60	1.23	−23.1
b2 (depth of holes 1, 2, 5, 6, and 7)	1.65	1.56	−5.45
b3 (length of hole 3)	1.80	1.43	−20.5
b4 (depth of hole 3)	1.65	1.72	4.24
b5 (length of hole 4)	1.00	0.96	−4.00
b6 (depth of hole 4)	1.65	1.55	−6.06

The design process presented successfully incorporates machining cost into a structural shape optimization problem. In addition to ensuring the manufacturability of the optimized components, the design process delivers components with a minimum cost and the required performance. The trade-off between structural performance and machining cost is critical, as demonstrated in this torque tube example.

19.7 SUMMARY

In this chapter, we discussed multiobjective design optimization, in which more than one objective is being minimized simultaneously. We started with basic concepts and Pareto optimality, and then we used a simple pyramid example to illustrate the essential points of multiobjective design optimization, especially the Pareto front in the criterion space. We introduced major solution techniques in four categories: methods with a priori articulation of preferences criteria, a posteriori articulation of preferences, no articulation of preferences, and multiobjective genetic algorithms. We pointed out the pros and cons of methods in individual categories and used simple examples for illustration. We also revisited the topic of decision-based design using decision theories, including utility theory and game theory. We reviewed the topic from the context of MOO problems and compared the decision methods with those of conventional MOO solution techniques. We pointed out that the solutions obtained using decision theories are subsets of the Pareto front. However, game theory supports a broader and practical design scenario that involves multiple designers or teams making design decisions simultaneously and sequentially. We reviewed commercial and academic codes for multiobjective design optimization and discussed their pros and cons.

In addition, we included advanced topics that are relevant to design optimization. We presented reliability-based design optimization that incorporates variations in physical parameters and design variables into design optimization. Also, we presented a case of design optimization that takes product cost, including manufacturing, as the objective function.

We hope that this chapter provides readers with adequate depth and enough breadth in these important and practical topics. We hope the materials presented in this chapter explained the basic concepts and solution techniques for solving MOO problems. We hope the overview on software programs offered ideas on the kind of codes that you may further investigate or adopt for solving the MOO problems you are encountering. Finally, we hope this chapter broadened your understanding of design theory and the methods in general, as well as added more practical tools to your toolset for engineering design.

QUESTIONS AND EXERCISES

19.1. In the design space, plot the objective function contours for the following unconstrained problem and identify the Pareto optimal set:

$$\text{Minimize: } f_1(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 3)^2$$
$$f_2(\mathbf{x}) = (x_1 - 5)^2 + (x_2 - 8)^2$$

Draw the gradients of each function at any point on the Pareto optimal set in the design space.

Comment on the relationship between the two gradients.

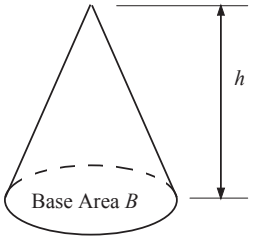
19.2. Sketch the Pareto optimal set for Problem 19.1 in the criterion space. Find the utopia point. Is the utopia point attainable?

19.3. A constrained optimization of two variables and two objectives is given below:

$$\begin{aligned} \text{Minimize: } & f_1(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 3)^2 \\ & f_2(\mathbf{x}) = (x_1 - 5)^2 + (x_2 - 8)^2 \\ \text{Subject to: } & g_1 = -x_1 - x_2 + 10 \leq 0 \\ & g_2 = -10 - 2x_1 + 3x_2 \leq 0 \end{aligned}$$

- **a.** Sketch the objective function contours and constraint functions in the design space and identify a feasible set S and the Pareto optimal set.
- **b.** Write a MATLAB script to find the Pareto front in the criterion space using the generative method. Graph the feasible and infeasible regions and identify the Pareto front in the criterion space.

19.4. A right circular cone shown below is considered for a redesign. The objective is to use a minimum material (surface area) to achieve a maximum volume. More specifically, we want to minimize both the lateral area S and total surface area T of the cone for a volume no less than 250 cm^3 by varying the base radius r and height h of the cone. Note that $T = S + B$, in which B is the based area $B = \pi r^2$. The size of the cone is constrained by $r \in (1, 10)$ cm and $h \in (1, 20)$ cm.



Volume = 1/3 Bh

- **a.** Formulate a MOO problem for the cone design mathematically.
- **b.** Sketch the objective function contours and constraint functions in the design space and identify a feasible set S and the Pareto optimal set.
- **c.** Solve the respective single-objective optimization problems to find the utopia point. Comment on the design of the cone at the utopia point.
- **d.** Sketch the Pareto front in the criterion space using the generative method. Graph the feasible and infeasible regions and identify the Pareto front in the criterion space.

19.5. Solve Problem 19.4 using the weighted-sum method. Find at least 10 solutions by varying the weights (including a case for $w_1 = 0.1$ and $w_2 = 0.9$). Identify those solutions in the Pareto front in the criterion space.

19.6. Solve Problem 19.4 using the weighted min–max method, assuming $w_1 = 0.1$ and $w_2 = 0.9$. Compare results with those of Problem 5. Comment on the pros and cons of the weighted

min–max and the weighted-sum method based on the observation made in solving the cone problem.

19.7. Solve Problem 19.4 using the NBI methods by creating at least five points on the CHIM.

19.8. Solve Problem 19.4 using the min–max method.

19.9. Derive Eq. 19.21c of Example 19.7: $\beta = \frac{1}{2}[(\alpha_1 + \alpha_2) + \sqrt{2 - (\alpha_1^2 - \alpha_2^2)}]$. Hint: $(\alpha_1 - \beta)^2 + (\alpha_2 - \beta)^2 = 1$.

19.10. Solve Problem 19.4 using the NSGA II of the genetic algorithm. Follow the discussion in Section 19.3.5.3 to download the MATLAB code and write two MATLAB files for the cone design problem. Use population size = 200 and number of generations = 500.

19.11. Solve Problem 19.4 using NIMBUS software online. Comment on the advantages and disadvantages of the software from a user's perspective.

REFERENCES

Arora, J.S., 2012. Introduction to Optimum Design. Academic Press, Waltham, MA.

Branke, J., Deb, K., Miettinen, K., Slowinski, R. (Eds.), 2008. Multiobjective Optimization: Interactive and Evolutionary Approaches. Springer-Verlag, Berlin, Heidelberg.

Choi, K.K., Kim, N.H., 2006. Structural Sensitivity Analysis and Optimization 1: Linear Systems. (Mechanical Engineering Series). Springer, Dordrecht.

Das, I., Dennis, J., 1998. Normal-boundary intersection: a new method for generating pareto optimal points in multicriteria optimization problems. SIAM Journal on Optimization 8 (3), 631–657.

De Weck, O.L., October 30–November 2, 2004. Multiobjective Optimization: History and Promise, the Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems (CJK–osm3) (Kanazawa, Japan).

Deb, K., 2001. Multiobjective Optimization Using Evolutionary Algorithms. John Wiley and Sons, Ltd, New York.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6 (2), 182–197.

Dieter, G.E., 1991. Engineering Design: A Materials and Processing Approach, second ed. McGraw Hill, New York.

Edke, M., Chang, K.H., 2006. Shape optimization of heavy load carrying components for structural performance and manufacturing cost. Journal of Multidisciplinary Structural Optimization 31 (5), 344–354.

Haug, E.J., Smith, R.C., 1990. DADS-Dynamic Analysis and Design System. Multibody Systems Handbook 161–179.

Hazelrigg, G.A., 1998. A framework for decision-based engineering design. ASME Journal of Mechanical Design 120, 653–658.

Jones, D.R., 2001. A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization 21, 345–383.

Kim, I.Y., de Weck, O., 2006. Adaptive weighted sum method for multiobjective optimization: a new method for Pareto front generation. Structural and Multidisciplinary Optimization 31 (2), 105–116.

Marler, R.T., Arora, J.S., 2004. Survey of multiobjective optimization methods for engineering. Structural Multidisciplinary Optimization 26, 369–395.

Messac, A., Mattson, C.A., 2002. Generating well-distribute sets of Pareto points for engineering design using physical programming. Optimization Engineering 3, 431–450.

Miettinen, K., 1999. Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston.

Poles, S., Vassileva, M., Sasaki, D., 2008. In: Branke, J., Deb, K., Miettinen, K., Slowinski, R. (Eds.), Multiobjective Optimization Software, Multiobjective Optimization Lecture Notes in Computer Science, vol. 5252. Springer-Verlag, Berlin Heidelberg, pp. 329–348.

Sawaragi, Y., Nakayama, H., Tanino, T., 1985. Theory of Multiobjective Optimization. In: Mathematics in Science and Engineering, vol. 176. Academic Press Inc, London, UK.

Schaffer, J.D., 1985. Multiple objective optimization with vector evaluated genetic algorithms. In: Grefenstette, J.J. (Ed.), Proceedings of an International Conference on Genetic Algorithms and Their Applications. Laurence Erlbaum Associates, pp. 93–100.

Srinivas, N., Deb, K., 1994. Multiobjective function optimization using nondominated sorting genetic algorithms. Evolutionary Computation 2 (3), 221–248.

Stadler, W., 1979. A survey of multicriteria optimization, or the vector maximum problem. Journal of Optimization Theory and Applications 29, 1–52.

Steuer, R., 1985. Multiple Criteria Optimization: Theory, Computation and Application. John Wiley & Sons, New York, NY.

Yu, X., 1996. Reliability and Durability Based Design Sensitivity Analysis and Optimization, Ph.D. Dissertation, The University of Iowa, Iowa City, IA, May 1996.

Yu, X., Choi, K.K., Chang, K.H., 1997. A mixed design approach for probabilistic structural durability. Journal of Structural Optimization 14, 81–90.

Index

Note: Page numbers followed by "b", "f" and "t" indicate boxes, figures and tables respectively.

A

ABAQUS, 12, 278, 725–726
Academic codes, 1145–1147
ACIS, 42, 138, 156–157, 161, 296–297
Acrylonitrile butadiene styrene (ABS) materials, 20, 751
ε-Active constraint strategy, 18, 18f
ε-Active strategy, 949–950, 952
  constraint strategy, 18, 18f
ADAMS, 11, 279, 434
Addendum surface designing, 716–718
Ad-hoc file management methods, 270–273
Adjoint load, 1022, 1022b
Adjoint response, 1027
Adjoint variable method, 1022. *See also* Direct differentiation method
  of continuum-analytical approach, 1050–1052
  for continuum approach, 1027–1029
  of continuum-discrete approach, 1046–1050
  for discrete approach
    adjoint load calculation, 1022b
    modulus design variable, 1023b
    performance measures, 1020b
Administration, product data management utility function, 282–283
Advanced mates, 176–177, 176t, 177f
Affine transformation, 100–102
AFGROW, 506
AFR (Automatic Feature Recognition) technology, 821
Aircraft torque tube, 1165, 1165f
  design
    optimization, 1168–1169
    parameterization, 1166
  FEA, 1167–1168
  optimization results, 1169f, 1169t
  problem definition, 1166
  VM, 1168
Airplane engine, 8f, 23–26, 234
  component-level design, 25, 25f
  design trade-off, 25–26
  design change propagation, 9f
  finite element meshes of connecting rod, 10f
  motion model, 11f
  rapid prototyping, 26
  sensitivity coefficients for performance and cost measures, 25–26
  single-piston, 8
  system-level design, 23–24
    design variable changes, 23–24, 24t
    load-carrying components, 24, 24f
    power, definition, 23
    stroke, definition, 23–24, 23f
  tubing, reverse engineering of, 112, 114f. *See also* Single-piston engine
Aluminum 2024 sheet, 734
  deflection due to springback, 706b
  principal stresses and sheet thickness, 696b
Analysis models, 8–10, 10f
Analytical derivative method, 1013–1015
Anisotropic yield criteria, 700–701
Anisotropy coefficient, 697
Annealed low-carbon steel sheet, 693
ANSYS, 11–12, 14, 30–32, 296, 978–980
  probabilistic Design System (PDS), 589
APriori's Product Cost Management, 822
Area turning sequence, 621, 622f
Arena Cloud PLM, 288–289
Art applications using RP system, 761–763
ArtCAM, 629–630
Assembly
  level, design parameterization at, 248–253
    guidelines, 248–249, 249f
    of HMMWV, 258, 260f
    of single-piston airplane engine, 256–257
    slider-crank assembly, 250–251, 251f–252f
  modeling, 169
    in CAD, 172–184
    case study and tutorial example, 225–228
    kinematic modeling technique, 199–225
    technique, 184–198
  Pro/ENGINEER, importing, 292–294
Asynchronous interactions, 303
Attributes, 850
AutoCAD, 161
Autodesk, 161
Autodesk Inventor, 161, 286t
Autodesk ProductStream, 286, 286t
AutoEditNC, 631
AutoForm, 722, 726
Automatic Feature Recognition (AFR) tools, 630
Automatic mesh generation
  AutoGen capability, 369
  clean-up, definition, 368

Automatic mesh generation (*Continued*)
mesh types, 365–369
quadrilateral meshes
direct quad and hex mesh, 367–368, 368f
indirect quad mesh, 366–368, 367f
MAT, 367
three-dimensional torque tube, 369, 369f
triangular and tetrahedral meshes, 365–366, 366f
Automatically Programmed Tools (APT), 608
advantages of, 609
language statements used, 609
profile milling example, 609f–610f
source codes, 610–611
Axioms. *See* Design axioms
3-axis mill, 633–634
with ball-nose cutter, CL data, 671–674
with flat-end cutter, 668–670
toolpath, 668–670, 669f
4-axis mill with flat-end cutter, 674–676
CL data, 675–676
rotary table, 605
toolpath generated, 675f
5-axis CNC mill, 605
5-axis mill with ball-nose cutter, 658–667
CL data, 662–666
distances between passes at edges, 658
lengths of the edges, 658
number of passes, 658–659
parametric representations, 662–666
scallop height, 659–661
Axis of motion, 603

B

Backward induction, 881
Ball joint, 180f, 181
Ball nose cutter, 625, 653
Bar example, 1005
differential equation, 1006–1007
energy equation, 1007–1009
finite element formulation, 1009–1012
Basquin's rule, 479–480
Batch-mode design optimization, 975
Bathroom transport device, 171, 171f
Bayesian approach, 861–862
Bayes' theorem, 861–862
BBC (Banabic-Balan-Comsa) criteria, 701
Beam displacements, 1042
Bearing and crank assembly, mating constraints for, 174, 174f, 197–198. *See also* Slider-crank mechanism
Beer can design, 911f
Bending stress, 1003
sensitivity, 1044–1046
BENSLOVE optimizer, 1145
Bernstein polynomials, 81, 1058–1059
cubic, 58f
quadratic, 53–54
Bézier curve, 62, 68, 100, 677–679, 1072, 1079
cubic, 53f, 56f, 62, 63f
scaling, 102–103, 102f
translation, 103–104, 104f
joining two curves, 63, 63f
quadratic, 47–56, 48f–49f
Bézier surface, 79f, 81–82
BFGS method. *See* Broyden–Fletcher–Goldfarb–Shanno (BFGS) method
Bicubic Bézier surface, 79f, 82, 1062, 1063f
Bicubic parametric surface, 1061–1062, 1078
Bicubic surface patch, 77–78, 79f
Bicycle wind measurement device (BWMD) using SEER-DFM, 832–837
bill of materials and costs breakdown, 834t
main window of SEER-DFM, 833
Bill of materials (BOM), 269, 276f, 286, 288
Binomial coefficient, 53
Bioengineering application using RP system, 768–771
Bisection search, 939
"The Black Bull" sign-making, 630, 631f
Blend surfaces. *See* Loft surfaces
Block
with center hole, 237f
design intent capturing for, 139, 139f
rectangular, boundary representation of, 135–138, 136f
Boeing 777
jetliner, 3
product data management system implementation, 284
BOMControl, 288–289
Boolean set operations, 131f, 132–135, 133f, 156, 244
Boundary representation (B-rep), 131, 135–138, 135f, 161, 244, 298
Boundary smoothing, 1095–1096
Boundary velocity computation. *See also* Domain velocity computation
implementation with CAD software, 1074–1075
2-D planar structures, 1068–1070
3-D solid structures
CAD-generated surfaces, 1072–1073
freeform surfaces, 1071
Bracket assembly, 237, 238f, 241, 241f
Broyden–Fletcher–Goldfarb–Shanno (BFGS) method, 944–946
Brute-force approach, 1116

B-spline curves, 64–74, 100, 108, 111, 244
basis functions of, 114–116, 116f
closed uniform, 71–74, 72f
fitting, 108–110, 109f, 112
nonuniform, 64–68, 65f, 67f–68f
uniform, 69–71
B-spline kernel function, 357, 357f
B-spline surfaces, 82–83, 83f, 111–112, 141, 1071
basis functions of, 114–116, 116f
construction for structural design, 112f
skinning, 110–112, 110f
BSM. *See* Bridging scale method (BSM)
Build plan, solid modeling, 157–160, 158f–160f, 242
Bull mill, 617–619
Business context, 849

C

CAD. *See* Computer aided design (CAD)
CAD/CAM approach, 611
CAD-centered CAM module, 630
CAD-generated surfaces, 1063–1065
CAD product model, 8
CAE. *See* Computer-aided engineering (CAE)
CalcMaster® Injection Molding Software, 823
CAM software. *See* Computer-aided manufacturing (CAM) software
CAMWorks, 7, 140, 161, 600
Cantilever beam, 1054, 1141–1143, 1141f
deformed cantilever beam, 1056f
discrete-analytical approach, 1055
displacement sensitivity, 1056
midpoint movement with design change, 1057f
shape design variable, 1055
shape sensitivity coefficients, 1057
simple cantilever beam, 991
PRO/MECHANICA structure, 995
SolidWorks simulation, 993
Carbide inserts, 624–625
Cartesian coordinate system, 78–79, 100–101, 103, 105–108, 605
circle in, 44–45, 45f
Cauchy strain tensor, 1032
Castigliano's theorem, 330
CatalystEX® software, 20
CATIA, 7, 161, 284, 286–287, 286t
CATIA CAD/CAM/CAE system, 629
CAVE (CAVE Automatic Virtual Environment), 225–227, 226f
applications for engineering design, 226f
CC-based toolpath generation methods, 648–649, 649f
CC curves, 1162–1163, 1162f
CDF. *See* Cumulative distribution function (CDF)
CD-R, 270
CE. *See* Concurrent engineering (CE)
CFD. *See* Computational fluid dynamics (CFD)
CHIM. *See* Convex hull of individual minima (CHIM)
Chord length, defined, 20
Chrysler's automotive interior design, 3
CIMCO Edit, 631
CL. *See* Cutter location (CL)
Classical methods. *See* Optimality criteria methods
CL data, 13
APT, 608
3-axis mill
with ball-nose cutter, 671–674
with flat-end cutter, 670
4-axis mill, 675–676
with flat-end cutter, 675–676
5-axis mill with ball-nose cutter, 658, 662–666
CAD/CAM approach, 611
cylindrical surface of Bézier curve, 678, 678f
for HAAS lathe, 621
inclined flat surface, 650–657
M-codes and G-codes for CNC machine, 611
questions related to, 666–667
ruled surface, 657–676
in Y- and Z-directions, inclined flat surface, 653f, 654
Closed-loop control, 603
Closed-loop kinematic system, 203–204, 203f, 207, 214–218
determining coordinate systems, 210f
link parameters for, 225t
z-axis and coordinate systems of joints, identification of, 210–211, 210f
Closed uniform B-spline curves, 71–74, 72f
Cloud-based service, 268
CNC (computer numerical control) machining, 19, 601, 605f
cover die machining, 22f
critical machining parameters, 625
cutters, 624–627
cutting speed, 626t
fixtures jigs, 623–624
machine control data (MCD), 22
machining parameters, 624–627
onsite adjustments, 625
practical aspects, 622–629
recommended feed per tooth for high-speed steel and carbide inserts, 627t
setting a sequence, 627–629
Simulator, 631
CNCezPro, 608, 608f
CNC machining. *See* Computer numerical control (CNC) machining

Codes
general purpose, 161
special, 161–162
Coefficient of normal anisotropy, 698, 703–704
Coefficient of planar anisotropy, 698
Coincident mates, 185–186, 187f
Combustion, 12
Commercial CAM software tools, 629–631
advantage of, 630
special-purpose software, 612f, 629–630
Commercial codes, 971
Commercial forming simulation software
AutoForm, 726
CATIA V5, 726
DynaForm, 727–730
Forming Technologies Inc. (FTI®), 726
HyperForm, 727
overview, 725–727
Pam-Stamp 2G, 726–727
Pro/ENGINEER, 726
SolidWorks, 726
Commercial tools, 1147–1150. *See also* Academic codes
Communication, product data management utility function, 282
Component-level design, for airplane engine, 25
changes, 26t
of connecting rod, 25, 25f
performance measures, 26t
variables, 26t
Composite scaling constant, 874
Composite transformations, 105–108
Computation of transformation matrix, 188–197, 188f, 190f, 197f
Computational fluid dynamics (CFD), 12, 14
Computer aided design (CAD), 4, 910, 1053, 1107
Computer-aided engineering (CAE), 4, 910, 1107
Computer-aided manufacturing (CAM) software, 4, 1071, 1107
Computer-aided methods
kinematic analysis
Cartesian generalized coordinates, 420
holonomic kinematic constraint equations, 416
particle kinematics formulation, 416b
planar mechanism, 416
slider-crank mechanism, 420, 420f
velocity and acceleration equations, 420
kinematic joints
compound joints, 422
constraint equation, 422–424
contact stress, 422
higher pair joints, 422, 423f
lower pair joints, 422, 423f, 424, 450t
planar J1 and J2 joints, 421, 422f
planar revolute joint, 422–424, 423f
multibody dynamic analysis
body-fixed reference frame, 425
DAE, 426–427
Lagrange multipliers, 425
variational equation of motion, 424–425
velocity and acceleration equations, 426b
nonlinear algebraic and differential equations, 415
Computer-assisted part programming, 608–611
Computer numerical control (CNC) machining, 2, 13, 19, 21–22, 22f, 35, 1161
cover die machining, 22f
machining, 21–22
Computer-aided manufacturing (CAM), 4, 43
ComputerVision, 285
COMSOL Multiphysics, 378
Concentric mates, 185–197, 187f
Concurrent engineering (CE), 3–4
Configuration change, 1053
Conjugate gradient method, 940, 943, 943b
Conic curves, 44, 45f, 117f–118f, 134
representing with NURB curves, 116–120, 119f–120f
Constrained optimization problems, 917
Constrained problems, 917, 949–964, 951f
ε-active constraint strategy, 952
algorithms, 950–951
feasible direction method, 957–962
feasible region, 950
linearized subproblem, 951
mathematical definition, 950
numerical methods, 950
penalty method, 962–964
SLP, 952–956
SQP algorithm, 956–957
strategies, 950
Construction features, 140–141
Constructive solid geometry (CSG), 132–135, 244
tree, 133–134, 133f, 249f
Continuities, 866
of Coons patch, 79–81, 80f
curve, 63, 63f
Continuum-analytical method, 1013
adjoint variable method, 1050–1052
Continuum approach, 1013, 1024
adjoint variable method, 1027–1029
direction differentiation method, 1025–1027
performance measure in, 1046–1047
sensitivity in, 1033
shape sensitivity analysis, 1087
for cantilever beam, 1088–1090, 1089b
material derivative, 1087–1088
variation operator, 1024

Continuum-discrete approach, 1013, 1029–1030
adjoint variable method, 1046–1050
Contour surface milling, 616, 620f, 648–649
3-axis mill for, 633–634
for a cylindrical surface, 677f
for a finish cut, 616–617, 620–621
for inclined flat surface, 650f
manufacturing parameters for, 633t
sculpture surface, 617, 617f
for tolerance 0.01, 678f
for tolerance 0.5, 679f
using 3-axis mill with ball-nose cutter, 653f
using Pro/MFG, 617
volume milling, 618
Conventional methods
decision matrix method, 850
aerospace applications, 851
airplane torque tubes, 850–851, 851f, 852t
material options, 852t
rating factor, 851–852
uncertainty, 853
weighting factors, 852, 853t
decision tree method, 853
decision making, 857
elements, 854
expected values, 856
OEM, 854–855, 854f
payoffs, 855
solutions to, 856f
state of nature, 854
Conventional product development, 2
cost/ECR *vs.* time, 3f
defects in, 6
Convex hull of individual minima (CHIM), 1126
Coons patch, 79–81, 79f, 1077
composite, continuity of, 80, 80f
cylindrical surface representation using, 120f
Cooperative games, 876, 883–884
Coordinates, homogeneous. *See* Homogeneous coordinates
Core panel, 731–732
Cosmo VRML Player, 278
Costs *vs.* time, 3f, 4, 5f
Cost estimates
accuracy, 789
aPriori's Product Cost Management, 822
for a BWMD using SEER-DFM, 832–837
CAD-based costing software, 820
commercial software, 820–823
Costimato®, 822
DFM Concurrent Costing, 821–822
direct *vs.* indirect, 797
elements, 793–794
fixed *vs.* variable, 795–797
fundamentals, 792–802
Geometric Cost Drivers (GCDs), 822
injection molding cost model, 810–816
machining cost calculation, 804–810
machining costing using SolidWorks, 824–829
manufacturing cost model, 802–819
MicroEstimating, 821
MISys® Manufacturing, 822–823
overhead costs, 797–798
qualitative techniques, 799–801
quantitative techniques, 801–802
SEER for Manufacturing (or SEER-DFM), 821
sheet metal stamping cost model, 816–819
sheet metal costing using SolidWorks, 829–832
special-purpose costing software, 823
techniques, 790, 798–802
types of cost, 794–797
Costimato®, 822
CostMate®, 823
Coulomb friction, 720
Coupled design intent, 236, 236f, 238, 239f, 248
Cournot duopoly model, 879–880
Cover die machining, 22, 22f
C++ program, 1163
Crack growth rate, 491, 491f
Criterion space, 1110–1111, 1110f, 1115f
Crosshatch options, 20, 22f
Crossover process, 966, 967f
Crowded-comparison operator, 1136
Crowding distance, 1136
CS0, 618
Cubic Bézier curve, 1058–1059, 1059f
Cubic curves, 56–62
basic functions of, 58f
Bézier curve, 53f, 56f, 62, 63f
B-spline curve, 71
Hermit cubic curve, 56f, 59–61
spline curve, 56–59, 56f
Cumulative density function (CDF), 529, 538f, 539
Cumulative distribution function (CDF), 538, 539f–540f
lognormal distribution, 545
Monte Carlo simulation, 549–552
Curved geometry drawbead design, 719f
Curves
fitting, 108–111, 109f
engineering applications, 111–112
parametric, 44–76, 44f–45f
Cut point, 966–967
Cutter location (CL), 13. *See also* CL data
Cutters, 624–627

Cutting speed, 625
for high-speed steel and carbide inserts, 624–625
recommended, 626t
CutViewer, 631
Cylindrical joint, 178, 180f
mapping mating constraints to, 201f, 202
origin and z-axis of, 220f, 221
Cylindrical surfaces, 84–85, 85f
Bézier curve, 677–679
tolerance, 678
representation using Coons patch, 81, 81f

D

DADS. *See* Dynamic Analysis and Design System (DADS)
DBD. *See* Decision-based design (DBD)
Data
exchange options, 291–292
transport and translation, 283
vault and document management, 273–274. *See also* Product data management (PDM)
Datum features. *See* Construction features
Decision-based design (DBD), 1141. *See also* Multiobjective optimization (MOO)
game theory, 1144
utility theory, 1141–1143
Decision criteria. *See* Attributes
Decision factor, 851–852
Decision making. *See* Design decision making
Decision-making models, 858–859
Decision matrix method, 850
aerospace applications, 851
airplane torque tubes, 850–851, 851f, 852t
material options, 852t
rating factor, 851–852
uncertainty, 853
weighting factors, 852, 853t
Decision node. *See* Terminal node
Decision theory, 857–858. *See also* Utility theory
decision-making models, 858–859
decision under risk, 859
car-buying example, 860–861, 861f
chance node, 860
cost of options and events, 860t
decision node, 860
expected payoff, 859–860
historical data, 859–860
decision under uncertainty, 861
Bayes' theorem, 861–862
car-buying example, 862, 863f
using posterior probability, 863, 863f
probability data, 863
elements, 858
cost of options and events, 858t
generic payoff table, 859t
Decision tree method, 853
decision making, 857
elements, 854
expected values, 856
OEM, 854–855, 854f
payoffs, 855
solutions to, 856f
state of nature, 854
Decomposability, 866
Decoupled design intent, 238, 248
Deep drawing process, 694f
Deform, 725
Degrees of freedom (DOFs), 170, 173–174, 174t, 176–177, 183–184, 200, 209
analysis, 197–198, 198f, 198t
of lower pair joints, 181t
Delaunay triangulation, 366
Dental application using RP systems, 767–768
Denavit–Hartenberg (D–H) representation, 202–218, 203f
closed-loop system, 214–218
illustration of parameters, 205f
link parameters, 207t
open-loop system, 210–212
Dentino-enamel junction (DEJ), 112, 353
Design
collaboration, 310
ENOVIA Smarteam, 286–287
paradox, 2, 2f
parameterization, 131
point, 1111
problem, 1159–1160
formulation, 15–16
process management, 14, 309–310
system-level, for airplane engine. *See* System-level design; for airplane engine
Design automated by computer (DAC-1), 126, 160
Design axioms, 238–242
independence axiom, 238–240, 239f–240f
information axiom, 241–242, 241f
Design-build-test process, 2–3
Design decision making, 15–19
parametric study, 16–17, 17f
problem formulation, 15–16
trade-off analysis, 17–18, 17f–18f
what-if study, 19
Design change propagation, 9f
Design criteria. *See* Attributes
Design decision making, 15–19
design problem formulation, 15–16

design trade-off analysis, 17–18
DSA, 16
feature-based design parameters, 16
parametric study, 16–17, 17f
what-if study, 19
Design for manufacturability (DFM), 4, 13
concurrent costing, 821–822
Design for manufacturing and assembly (DFMA), 4, 13
Design for X-abilities (DFX), 3
Design intent (DI), 139, 139f, 235–237, 236f, 244, 249
of single-piston airplane engine, 253–257, 254f
Design optimization, 909–910, 923–924
beer can design, 911f
case studies, 980–990
constrained problems, 949–964
deterministic, 1155–1156
engineering design problems, 910–911
gradient-based approach, 936–949
graphical solutions, 930–936
non-gradient-based approach, 964–970
optimality conditions, 918–930
practical engineering problems, 970–978
problem, 913–918
process, 1159–1160, 1160f
software, 978–980
tutorial example, 991–995
Design optimization tool (DOT), 971, 1156
Design paradox, 2, 2f
Design parameterization, 234
assembly level, 248–253
case studies, 253
design axioms, 238–242
design intent, 235–237
part level, 242–248
Design problem formulation, 15–16
Design requirements (DR), 235, 236f
Design sensitivity analysis (DSA), 16, 972–973, 1002, 1158, 1162–1163. *See also* Shape sensitivity analysis; Topology optimization
approaches for, 1014f
bar example, 1005
differential equation, 1006–1007
energy equation, 1007–1009
finite element formulation, 1009–1012
design variables, 1004f
formulation, 1004–1005
sensitivity analysis methods, 1012
analytical derivative method, 1014–1015
continuum approach, 1013, 1024–1029
discrete approach, 1017–1024
finite difference method, 1015–1017
sensitivity formulation, 1013
Design sensitivity coefficients, 975
Design space, 1110–1111, 1110f
Design Theory and Methods using CAD/CAE, 4, 18
Design trade-off
analysis, 17–18
method, 25–26, 32–35
Design variables (DVs), 236, 236f, 239, 240f, 242–244, 249
of HMMWV, 257f
of single-piston airplane engine, 255–256, 255f
Design velocity field computation, 1065, 1096–1097
boundary velocity computation
implementation with CAD software, 1074–1075
2-D planar structures, 1068–1070
3-D solid structures, 1071
changing of structural domain, 1066f
design sensitivity coefficients, 1067–1068
domain velocity computation, 1075
boundary displacement method, 1075–1076
isoparametric mapping method, 1076–1080
linearity requirement, 1067
in shape optimization, 1067
structural boundary change, 1067
DesignXplore response surface technologies, 979–980
DeskCNC, 630
Deviatoric stresses, 695
DFM. *See* Design for manufacturability (DFM)
DFMA. *See* Design for manufacturing and assembly (DFMA)
DFX. *See* Design for X-abilities (DFX)
Die-casting process, 632
cover and ejector dies, 633f
cutters used for volume milling, 633, 633t
machining operations of, 633
roadarm green part, 635–636
Difference operations, 132–133, 133f
Differential-algebraic equations (DAE), 426–427
Differential equation, 1006–1007
Dimension 1200 sst®, 19–20, 21f
Dimension 3D Printers from Stratasys, 751
Direct differentiation method, 1018. *See also* Adjoint variable method
for continuum approach, 1025–1027
of continuum-discrete approach, 1043–1044
for discrete approach
chain rule, 1019b
performance measure sensitivity, 1020
sensitivity coefficients calculation, 1020b
Direct model translations, 292–296
data exchange between CAD and CAE/CAM, 296
Pro/ENGINEER
importing assembly to SolidWorks, 294–295
importing parts to SolidWorks, 295

Direct model translations (*Continued*)
SolidWorks
importing assembly to Pro/ENGINEER, 294–295
importing parts to Pro/ENGINEER, 295
Direct modeling, 155
Discrete approach, 1017
adjoint variable method, 1022–1024
direct differentiation method, 1019–1021
discrete-discrete approach, 1019
fictitious load, 1017–1018
stiffness matrix and load vector, 1018–1019
Discrete optimization problem, 917
Displacement function, 1040
D-MEC's Solid Creation System (SCS), 751
DNC (distributed numerical control), 604
Documents, product data type, 275–276
Domain
of evaluation, 1061
shape change, 1053
shape sensitivity analysis, 1053–1054
Domain velocity computation, 1075. *See also* Boundary velocity computation
boundary displacement method, 1075–1076
isoparametric mapping method, 1076–1080
Dominated space, 1112
DOT. *See* Design optimization tool (DOT)
Drawbead design, 718
Draw bending test, 704, 704f
Draw forming (or deep draw), 689–690
blank holder (or binder) clamps, 689
categorization of, 690
tooling motions, 690
trimming operation, 689
Drawing, engineering, 126–127
DSA. *See* Design sensitivity analysis (DSA)
DynaForm, 687, 722, 725, 727–730, 729f
Dynamic Analysis and Design Systems (DADS), 10–11, 29, 279, 971
codes, 434
Dynamic simulation model for HMMWV, 29f
Dynamic stress calculation
"blocks to failure", 504–505
dynamic stress histories, 498
external forces, 497–498
fatigue life process, 497, 497f
inertia body force, 499, 499f
joint reaction forces and torques, 498
peak-valley editing, 500–501, 502f
quasistatic equation, 498
radial and tangential accelerations, 499–500
rain-flow counting
constant-amplitude loading, 501, 503f
hysteresis loops, 501, 503f
Miner's rule, 477
stress history, 497–498
variable-amplitude loading history, 481
stress influence coefficients, 498–499
superposition principle, 500
vehicle suspension components, 515
Dynamic workflows, 279

E

EA. *See* Evolution algorithms (EA)
Eccentricity, defined, 117
ECR. *See* Engineering change request (ECR)
e-Design paradigm, 2, 4–7, 6f
for airplane engine, 23–26
CAD, 5–7
concepts and methods for product development, 4
cost/ECR vs. time, 3, 3f, 5f
cycle time vs. cost/ECR and product knowledge, 5f
design paradox, 2, 2f
design variables, 6
environment for mesh generation, 8–10
features, 35
hierarchical product models evolved through, 15f
for high-mobility multipurpose wheeled vehicle, 26–35
IT-enabled technology, 2
KBE system, 5
paradigm, 5–7, 6f
physics-based simulation technology, 6
physical prototyping, 19–22
fabrication, 19
rapid prototyping. *See* Rapid prototyping (RP)
potentials of, 2
product design and physical prototype, 6, 6f
product development cycle, 2
product knowledge vs. e-Design cycle time, 4, 5f
Pro/ENGINEER and SolidWorks, 7
software environment, 4
tool integration for, 305–310
using CAD systems, 7
virtual prototyping, 7–19
Education and research application using RP system, 766
Effective stress, 695
for isotropic material at yielding, 696
Elastic deformation, 692, 698
Electron beam melting (EBM), 755–756
Elite points, 1133
Elitism, 1137
Elitist strategy, 1133
Ellipse, 117–118, 118f

Email, 282
  file attachments, 270
Empirical stress–strain laws, 694f
End-of-block character, 606
Energy bilinear form, 1033, 1041
  adjoint response and virtual adjoint response, 1047
  in material derivatives, 1082
  variations of, 1036–1037
Energy equation, 1007–1009
  for cantilever beam, 1038–1039
Energy method
  Castigliano's theorem, 330
  displacement calculation, 332b
  strain energy, 331, 331f
Engine block of a single-piston engine, 773–774
Engine case models, 21f
Engine connecting rod shape optimization, 986–987, 990f
  design optimization, 990
  design parameterization and problem definition, 989–990
  geometric and finite element models, 988–989
Engine motion model, 11f
Engineering
  data models, 275–280
    process data model, 279–280
    product data model, 275–279
  design, decisions in, 848–849
    conventional methods, 850–857
    decision theory, 857–864
    design examples, 885–901
    game theory, 875–884
    utility theory, 864–875
  drawing, 126–127
  views, 14
  engineering change requests (ECRs), 3
    vs. time, 3f, 4, 5f
  tool wrappers, 308–309
Engineering change request (ECR), 3
Engine motion model, 11f
Engraving machine, toolpaths for, 630
ENOVIA, 286–287
  Smarteam, 286–287, 286t
    enterprise collaboration, 287
Enterprise product data management, 285
Equivalent von Mises stress, 475
Equal interval search method, 937–938, 937f
Euclidean transformations, 100
Euler equations, 412
Euler-Poincare law, 137, 137f
Evaluation function, 1061
Explicit form of curve representation, 44, 44f
Explicit method (or dynamic explicit method), 720
EXPRESS schema, 297
Extended finite element method (XFEM), 12, 484
  branch functions, 494, 495f
  crack growth calculation, 492, 493f
  level set method, 494, 495f
  nodal enrichment in, 494, 494f
  vector-valued function, 493–494
  zigzag crack path, 496–497

F

Fabricating prototypes, 2–3
Fabrication processes, categories, 745
3D Fabulous Fashion Show, 762
Failure event/function, 12
Failure probability, 1152
  deterministic design *vs*. probabilistic prediction
    bending stress, 527, 531
    length parameter, 527
    material yield strength, 527
    MATLAB, 529
    mean value and standard deviation, 530, 530t
    PDF, 528–529
    probability distributions, 528, 528f
    solid circular cross-section, cantilever beam, 526, 527f
    standard normal distribution function, 529, 529f
    stress failure mode, 564
    yield strength, 531
  probabilistic design
    absolute-worst-case approach, 533
    cantilever beam, safety factor, 532, 533f
    norminv(p) function, 532
    yield strength, random variable, 532
  product performance, 534–535
FastForm, 726
Fatemi-Socie model, 483
Fatigue/fracture analysis, 12
  Aloha Airlines Flight, 465, 465f
  component circumference, 467, 467f
  crack initiation, 505–506, 511f
    general-purpose codes, 505–506
  crack propagation, 466
    FEA based approach, 507
    non-FEA based approach, 506–507
  crankshaft
    fatigue life of, 517–518, 517f
    finite element model, 516–517, 517f
    slider-crank mechanism and geometric dimensions, 516–517
  cumulative damage. *See* Dynamic stress calculation
  cyclic loads, 469, 469f
  engine connecting rod
    crack propagation analysis, 513–515, 515f–516f
    finite element model, 511–512, 512f–513f

Fatigue/fracture analysis (*Continued*)
material properties, 511–512
maximum principal stress contour, 512, 512f
mesh refinement, 513, 513t, 514f, 516f
residual life results, 515, 517t
second-order curve and polynomials, 515
fatigue, definition, 467
fracture mechanics. *See* Fracture mechanics
geometric stress concentration factors, 466–467
for HMMWV roadarm, 30, 32, 32f
DADS, 509
dynamic simulation, 509, 510f
fatigue life contour, 511, 511f
finite element model, 509–510, 510f
joint reaction forces, 509, 510f
static von Mises stress contour, 511, 511f
structural finite element model, 508, 508f
Liberty ships, 466
prediction analysis, 468
S-N analysis, 468
stages of, 464, 466
strain-based approach
low-cycle fatigue, 478
Manson–Coffin equation. *See* Manson–Coffin equation
multiaxial analysis, 483–484
stress-life approach. *See* Stress-life approach
structural components, 464
uniaxial stress, 469
United Airlines Flight, 465f, 466
Fault-tree analysis, 13
FDM. *See* Fused deposition manufacturing (FDM)
FEA. *See* Finite element analysis (FEA)
Feasible designs, 915–916
Feasible direction method, 957–962
constraint equations, 959–961
geometric description, 958f
linear programming problem, 959–961
objective function, 957–958
optimization problems, 958, 959b
push-off factor, 959–961
usability and feasibility requirements, 958
Feasible region. *See* Feasible set
Feasible set, 913
Feature-based design parameters, 16
Feature data exchange (FDE), 289–290
Feature recognition (FR), 299–301
Feature-based parametric solid modeling, 131, 131f, 139f, 234
direct modeling, 155
geometric features, 131
geometric modeling kernels, 155–157, 156f, 157t
parametric modeling, 150–151
parent–child relationships, 149–150, 150f
procedure in CAD, 151–154, 152f–154f
sketch profiles, 142–149
FeatureWorks, 300–301
Feedrate, factors affecting, 625–626
Feed table, 623
Feeler gage, 628–629
FEMFAT, 506
Fictitious load, 1017–1018, 1025
Files
management, 269–274
ad-hoc methods, 270–273
PDM approach, 273–274
product data type, 275–279
Finite difference method, 16–17, 25–26, 1015–1017, 1080–1082
shape sensitivity analysis using, 1080–1082
Finite element
discretization, 1040–1042
formulation, 1009–1012
meshes, 10f
Finite element methods/analysis (FEMs/FEA), 8–12, 14, 16, 129, 130f, 141, 340–359, 686–687, 1002, 1148, 1167–1168, 1167f
SolidWorks Simulation boundary value problem, 345
CAD model translations
boundary and loading conditions, 372–373, 373f
IGES standard, 297
NX–Patran interface, 377
STEP, 297
tracked vehicle roadarm, 1095
cantilever beam, 374, 374f, 380–382, 381f–382f
classical beam theory, 375–376
commercial software
ANSYS, 377
COMSOL Multiphysics, 378
LS-DYNA, 378
MSC/Abaqus, 377
MSC.Marc FEA program, 378
MSC/Patran, 377–378
Nastran, 377
Pro/ENGINEER, 377–378
SolidWorks Simulation, 377
cubic shape function, 381–382
degrees of freedom, 346–347
Dirac delta function, 341
domain discretization, 345–346, 346f
element meshes of a connecting rod, 10f
element shape functions, 349
element types, 351, 352f
2D engine connecting rod, 350–351, 351f
equilibrium equation, 342

error estimation, 350
essential boundary conditions, 341
fidelity/efficacy, 376
finite element modeling
  automatic mesh generation. *See* Automatic mesh generation
  conformal and nonconformal mesh, 363–371, 364f
  geometric features, 361
  geometric idealizations, 361, 362f
  geometric simplification, 361–362, 362f
  linear elastic range, 360–361
  model creation process, 360, 360f
  Nastran, 359
  semiautomatic mesh generation, 365. *See also* Semiautomatic mesh generation
  SolidWorks Simulation, 363
  thin-walled tank problem, 363, 364f
force vector, 343–344
four-node quadrilateral mesh, 346–347, 347f
fourth-order polynomial function, 345
frequency response analysis, 380
function discretization, 346, 347f
geometric model, 358
h- and p-methods, 340
for HMMWV, 32, 32f
human middle ear anatomy, 378, 379f
HyperMesh, 380
interpolation functions, 342
Jacobian matrix, 347–348
maximum bending stress, 375
meshless method
  RKPM. *See* Reproducing kernel particle method (RKPM)
  SPH method, 356
mesh refinement and convergence, 376
middle ear finite element model construction, 379
natural boundary conditions, 342
Newton's second law, 375
origin of, 340
partial differential equations, 340
Poisson's ratio, 375
process of, 358, 359f
p-version FEA
  advantages, 353
  DEJ, 353
  displacement interpolations, 351–352, 352f
  energy norm, definition, 353
  hierarchical shape functions, 353
  linear static analysis, 355
  Pro/MECHANICA Structure, 353–354, 354f
  stress fringe plots, 355, 355f
  tooth model, 354f, 355
  tooth stress distribution, 355, 355f
  reduced matrix equations, 344
  simple cantilever beam, 341, 341f
  stiffness matrix, 343–344, 349
  strain field computation, 347–348
  stress error, 350
  stress jumps, 349, 349f
  structural, 25
  thin-walled tube
    half-tank model, 363, 364f
    tube surface model, 383–384, 383f
  unit systems, 374
  virtual displacement, 341
Finite element reliability analysis (FE-RA) software, 588
Finite Element Reliability Using MATLAB (FERUM), 589
First-order reliability method (FORM), 13, 1152
  average correlation coefficient approximation, 587
  De Morgan's law, 586–587
  Ditlevsen's bounds, 587
  failure element, 585
  FEM, 552
  Gollwitzer and Rackwitz's approximation, 587
  Hohenbichler's approximation, 587
  joint PDF, 553
  linearized limit state functions, 586, 586f
  MPP search algorithm, 555, 557–559, 566f
  PMA, 563–564, 564f
  random variable transformation, 553, 564f
  RIA, 559–560, 582f
  simple bounds, 587
  standard normal distribution, 553, 555–557
First-order Sobolev space, 1008
Fitness function, 966
Fitness sharing, 1133
Fitting. *See* Curves; fitting
Fixtures, 623–624
Flow criterion, 698
Flow curve equation, 693
Ford C3P, product data management system implementation, 284
FORM. *See* First-order reliability method (FORM)
Forming limit curve (FLC), 702
  based on Keeler and Goodwin law, 702
  factors affecting, 702–703
  for mild steel, HSS (high strength steel), and selective AHSS, 702
  using simulation package Pam–Stamp, 703–704, 703f
Forming limit diagram (FLD), 688, 701–704
Forming Technologies Inc. (FTI®), 726

Formula SAE racecar model
  camber angle, 445, 451f
  external force, 441, 442f
  modified profile cam, 441, 444f
  Pro/ENGINEER model, 439
  quarter suspension components, 436–437, 438f
  racecar-style suspension, 435, 437f
  result graphs, 433, 441f
  result verification, 439, 442f
  right front quarter, 436, 438f
  rigid joints, 437, 440f
  road profile, 437, 439f
  rocker shape change, 444–445, 448f
  segment velocity, 441, 445f
  shock travel, 441, 443f, 444–445, 446f–447f
  spring and damper, 441, 442f
  spring position, 441, 443f
Fracture analysis, 12
Fracture mechanics
  crack nucleation, 484
  damage tolerant design and analysis approach, 484
  energy approach, 485
  energy release rate, 485, 485f
  LEFM, 484
    fracture toughness, 486
    geometric factor, 487
    J-integral, 487–488
    path-independent closed contour, 487, 488f
    SIFs, 486
    Westergaard stress function, 487
  mixed mode
    crack tip opening modes, 488, 488f
    J-integral, 490
    mode 1 SIF, 490
    mode 2 SIF, 491
    normal and shear loads, 489–490, 489f
    stress element, 489–490, 489f
  plane strain problems, 486
  quasistatic crack growth, 491–492, 491f
  stress intensity approach, 486
  XFEM, 484
    branch functions, 494, 495f
    crack growth calculation, 492, 493f
    level set method, 494, 495f–496f
    nodal enrichment in, 494, 494f
    vector-valued function, 493–494
    zigzag crack path, 496–497
Fracture toughness, 485
Frechet distribution, 546
Freeform
  parametric surfaces, 1061
  surfaces, 1061–1063, 1071
Freiman curve, 789–790, 790f
Frenet frame, 96
Fused deposition manufacturing (FDM) technology, 20
FTP, 270

G

Game theory, 875, 1144. *See also* Utility theory
  cooperative games, 883–884
  design decisions, 875
  as design tool, 894, 901
    cooperative game, 900–901, 901f
    design solutions in design space, 898f
    Nash solution, 899f
    pressure vessel design example, 894–895
    sequential game, 897–900, 900f
    strategy form game, 896–897
  elements, 876–877
    Nash equilibrium, 877
    payoff table, 876t
  sequential games, 880
    backward induction, 881
    game tree, 877, 880f
    Nash equilibrium, 881
    solution of, 881
    Stackelberg duopoly model, 882
    strategy profile, 881
  two-person matrix games, 877–878
    Cournot duopoly model, 879–880
    Nash equilibrium, 878
    payoff table, 878t
    real numbers and payoff functions, 879
    system of equations, 879
G-codes, 21–22, 608
GA. *See* Genetic algorithm (GA)
Gear hub assembly models for HMMWV, 28f
General Motors' locomotive engine, 3
General plane stress sheet processes, 693–697
General-purpose machining software, 629–630
Generative method, 936, 937f, 965
Generic payoff table, 859t
GENESIS software, 980
Genetic algorithm (GA), 965
  design representation, 965–966
  genetic operations, 966–968
  mimicking evolution principle, 965
  reproduction process, 966–968
  selection, 966
  solution process, 968
Genetic drift, 1133
Geometric data exchange (GDE), 289–290
Geometric features, 131, 138
  classification of, 140f
Geometric model files, 275–276

Geometric modeling, 42
CAD-generated surfaces, 84–99
case studies, 108–112
kernels, 155–157, 156f, 157t
parametric curves, 44–76
parametric surfaces, 76–83
transformations, 100–108
software, 1077
Geometry
library. *See* Geometric modeling; kernels
mating, 173. *See also* Mating constraints
GibbsCAM, 629–630
Global
force vector, 1041
minimum function, 919, 920f, 923
stiffness matrix, 1041
Global design trade-off, 32–35
Golden section search method, 938, 938f
Goodman line criterion, 472–473
Gouge checking, 679
Gradient-based approach, 910, 918, 936–949. *See also* Non-gradient-based approach
convergent criterion, 936
generative method, 936, 937f
gradient-based search, 939–946
line search, 946–949
optimization method, 1029
search methods, 937–939
Gradient-based search, 939–946. *See also* Line search
BFGS method, 944–946
conjugate gradient method, 943, 943b
convergence criteria, 939
objective function, 939b
quasi-Newton method, 944
steepest decent method, 940–942
Gradient descent method. *See* Steepest decent method
Gradient(s), 975
calculations, 971
gradient-based algorithms, 1003
operator, 1047, 1083
vector, 940
Granite One, 156
Graphical solutions, 930–936
LP problems, 931–932
NLP problem, 933–936
Grashof's law, 181, 408–409
Gruebler's count, 430
Gumbel distribution function, 546

H

Haptics, 225–226
HAAS 3-axis mill, 600, 605
HAASmini-mill, 627
HAAS VF-series, 22
Hermit cubic curve, 56f, 59–61, 1078
joining two curves, 63, 63f
Hessian matrix, 566, 921–922
Hierarchical product model for HMMWV, 15f, 27–28
Higher feed per tooth, 626–627
Higher pair joints, 178
High-mobility multipurpose wheeled vehicle (HMMWV), 4–5, 15f, 26–35, 275–276, 525, 971
ANSYS analyses, 30–32
boundary and loading conditions, 372, 373f
design parameterization of, 253, 257–261, 257f
design change, 261, 261f
relations defined for differential, 259t
relations defined for steering rack, 259t
track design variable, 258–260, 258f–260f
wheelbase design variable, 257f, 258, 261f
design trade-off, 32–35
detail design, 28f, 30–32
driver seat vertical accelerations, 31f
dynamic simulation model, 28–29, 29f–30f
fatigue and fracture analysis, 33
DADS, 509
dynamic simulation, 509, 509f
fatigue life contour, 511, 511f
finite element model, 509–510, 510f
joint reaction forces, 509, 510f
static von Mises stress contour, 511, 511f
structural finite element model, 508, 508f
gear hub assembly models, 28f
hierarchical product model, 27–28
lower control arm models, 29f, 32f
low-fidelity CAD solid model of, 28–29
motion analysis
absorbed power equations, 449
constraint functions, upper bound, 450, 451t
design optimization of, 450, 451t
design problem, 447–448
front suspension, dynamic simulation, 448, 452f
initial and optimal design, 450, 451t
vehicle and suspension assembly, 445–446, 447f
vehicle performance, 450, 453f
overall objective, 26
preliminary design, 26–29, 27f
reliability analysis
CDF graphs, 584f, 590–591
crack initiation life prediction, 589–590, 590t
fatigue life contour, 557f, 589
FE model and geometric parameters, 557f, 589–590
rigid-body dynamic simulation, 588

High-mobility multipurpose wheeled vehicle (HMMWV) (*Continued*)
sensitivity of load on the spherical joint of control arm, 34f
shock absorbers, 30f–31f
forces, 31f
operation distance, 30f
suspension design, 27
virtual prototyping
buckling load factor, 30–32
control arm, design parameters, 32, 33f
detailed product model, 28, 28f
driver seat vertical accelerations, 29, 31f
dynamic model, 28–29, 30f
dynamic simulation, 28–29, 30f
gear hub assembly models, 28, 28f
lower control arm models, 28, 28f, 32, 32f
preliminary design model, 27, 27f
sensitivity coefficients, 32, 34f
shock absorber force history, 29, 31f
shock absorber operation distance, 29, 30f
spring constant, 27
Hill-climbing moves, 969
Hill's yield criterion, 700
Hinge joint, 178, 180f
Hole-drilling machine, 604
Hollomon's law, 693
Homogeneous boundary conditions, 1009
Homogeneous coordinates, 100–102, 101f, 188, 203–204
Horizontal stretch forming process, 690
Human maxillary second molar modeling, 112, 113f
Human middle ear modeling, 112, 113f
h-version finite element analysis, 10, 10f
Hydrostatic stresses, 695
Hyperbola, 118, 118f
HyperForm, 687, 727, 728f
HyperMesh®, 8–10, 276–278
HyperText Markup Language (HTML), 276–278

I

Identifier, 606
Image services, 283
iMAN, 285
Implicit form of curve representation, 44, 44f
Implicit method (or static implicit method), 720
Importance sampling method, 575–578, 576f
Inactive constraint, 927
Inclined flat surface, 650–657
CC lines and CL points, 650–651
contour surface milling for, 650f
offset points, estimating, 653
parametric method, application of, 649
scallop height, 652, 654, 655f
toolpath, 651
Independence axiom, 238–240, 239f–240f, 255–256
Independent principal vector (IPV), 197, 198f, 200
IND-NIMBUS system, 1147
Inertial frame, 203, 221–222
Infeasible solution, 916
Information axiom, 241–242, 249, 256
minimization of information contents, 241f
Initial Graphics Exchange Standards (IGES), 275–276, 290–291, 372
file structure and data format, 311–316
directory entry section, 312
global section, 312
parameter data section, 312–315
start section, 312
terminate section, 315–316
Injection molding cost model, 810–816
machine rate, 811–813, 812t
material data, 812t
molding cost estimate, 810
molding cycle time, 813–814
Instances, 132
Instantaneous area of the test specimen, 698
Integer programming problem. *See* Discrete optimization problem
Integrated Design and Engineering Analysis Software (I-DEAS), 161
Integrated product and process development (IPPD), 3
Integrated testbed using Windchill, 303–305
Intent. *See* Design intent (DI)
Interactive design process, 975–978
sensitivity display, 975–976
trade-off determination, 977–978
what-if study, 976–977
International TechneGroup Inc., 298
Intersection mating geometry (IMG), 198, 198f, 221–222
Intersection of primitives, 133, 133f
surface to surface, 134, 134f
Investment casting, 19
IPro 8000, 750
IPro 9000, 750
IPro 8000 MP, 750
IPro 9000L, 750
Iso-parametric machining, 649
Isosceles trapezoid nesting, 714
Isotropic von Mises yield criterion, 695, 699, 705
for plane stress, 699, 699f

J

Jigs, 623–624
jMetal optimizer, 1145

Joint
coordinate systems, 219–225
kinematic. *See* Kinematic joints
mating, 173. *See also* Mating constraints
selection in motion simulation models, 10–11
probability density function, 12
JT files, 271
JT Viewer, 272

K

Karush–Kuhn–Tucker (KKT) conditions, 927–930
active and inactive constraint, 927
Lagrange multipliers, 927
necessary conditions, 928
optimality conditions, 928–930
optimization problem, 928b, 930
slack variable, 927
Keeler–Brazier equation, 702
Keeler–Goodwin diagram, 701–702
Kernels, geometric modeling, 155–157
Kinematic joints, 178–184, 180f
compound joints, 422
constraint equation, 422–424
contact stress, 421–422
higher pair joints, 422, 423f
lower pair joints, 422, 423f, 424, 450t
mapping mating constraints to, 200–202, 201f, 202t
planar J1 and J2 joints, 421, 422f
planar revolute joint, 422–424, 427f
Kinematic modeling, 172, 199–225, 199f–200f
constructing joint coordinate systems, 219–225
Denavit–Hartenberg representation, 202–218, 203f, 205f, 207t
mapping mating constraints to kinematic joints, 200–202, 201f, 202t
Knowledge, product, 5f
Knowledge-based engineering (KBE), 4–5

L

Lankford parameter, 697
Lagrange multipliers, 424–425, 924–927
Lagrangian function, 925
Laser Engineered Net Shaping (LENS) of Sandia, 757
2½ D layers, 19, 20f
Level set method (LSM), 494, 495f–496f
Levy–Mises flow rule, 695
Lexicographic method, 1124–1125
L'Hôpital's rule, 870
Life cycle
design, product, 3
product life cycle management, 266–267
Limit state function, 1152
Line, straight, 46–47, 47f
Linear elastic fracture mechanics (LEFM), 486–488
fracture toughness, 486
geometric factor, 487
J-integral, 487–488
path-independent closed contour, 487, 488f
SIFs, 486
Westergaard stress function, 487
Linear elasticity
infinitesimal strains/"small" deformations, 333
stress components, 333–334, 333f, 335b
stress function, 334
Linear strain distribution, 705–706, 705f
Line search, 939–940, 946–949
objective function value, 946
optimization problem, 947–948
secant method, 948–949
Linear optimization problems, 917
Linear programming (LP) problems, 931–932, 931b
Liquid-based RP systems, 750–751
LMS® SYSNOISE, 11–12
Load linear form, 1033
in material derivatives, 1082
variations of, 1036–1037
Local design trade-off, 33
Local minimum function, 919, 920f, 923
Local strain (ε-N) approach, 468
Loft surfaces, 89–92, 90f
Lower control arm models for HMMWV, 29f, 32f
Lower pair joints, 178, 180f, 181t
DOFs of, 181t
LP problems. *See* Linear programming (LP) problems
LS-Dyna, 11–12, 376–377, 725–726
LS-OPT tool, 980
LSM. *See* Level set method (LSM)

M

Machine code data/machine control data (MCD), 22, 606
manual programming at, 608
Machining cost model, 804–810
material cost, 810
non-operation time, 809
operation time, 806–808
rough cutting, time for, 807t
setup time, 805, 805b, 806t
surface generation rate, 806–808
tooling cost, 809–810
Machining features, 140
Machining sequences, 612f
Manifold objects, 136–137, 136f
Manual NC programming, 606–608
Manufacturing applications using RP system, 761

Manufacturing cost model, 802–819, 1160–1161
  cost elements for in-house parts, 803–804
  processing cost, 803–804
  production time for a unit, 804
Manson–Coffin equation
  Basquin's rule, 479–480
  Miner's linear damage accumulation rule, 481
  Neuber's rule, 481, 482f
  plastic and elastic strain-life curve, 480, 481f
  slip band, elastic material, 479, 480f
  strain-based fatigue life prediction, 478–479, 479f
Manufacturing
  features, 140
  issues, 3
Mapkey, 1164, 1165f
Mass-IH process, 758–759
MasterCAM, 127, 128f, 129, 130f, 292, 612–613, 630
Material derivatives, 1082, 1083f
  deformed cantilever beam, 1084f, 1084b
  design velocity field, 1086
  of displacement, 1086
  energy bilinear form, 1082
  FEA solution, 1084
  gradient operator, 1083
  load linear form, 1082
Material features, 140
Mating constraints, 173–178, 174f, 174t, 197, 219
  aligned and antialigned conditions, 175f
  mapping to kinematic joints, 200–202, 201f, 202t
MATLAB, 85, 88–89, 88b, 1114
  functions, 1070f
  implementation, 1137–1141
MatrixOne, 268, 285
MAU. *See* Multiattribute utility functions (MAU)
Maxwell–Huber–Mises criterion, 699
MCD. *See* Machine control data (MCD)
M-codes, 21–22
MD. *See* Molecular dynamics (MD)
Mean time between failure (MTBF), 13
Mechanica. *See* Pro/MECHANICA structure
Mechanism Design, 10
Medial axis transformation (MAT), 367
MEMS (microelectro-mechanical system), 744
Metadata, 273
Meta-database, 273
Metal forming simulations, 688–689
Metaphase, 285
MicroEstimating, 821
Micro-manufacturing RP systems, 757–759
Micro-SLA systems, 758
Microsoft Excel, 16–17, 17f
Microsoft SharePoint, 270–271
Middle ear modeling, 112, 113f
Military, use of virtual reality in, 227, 227f
Milling machine vise, 624, 624f
Miner's rule, 477
Mirror features, 141, 142f
MISys® Manufacturing, 803–804
modeFRONTIER®, 1148
  third-party applications, 1149t
  workflow editor in, 1149f
Model fabrication using RP system, 780–781
Modeling engine. *See* Geometric modeling; kernels
MOGA. *See* Multiobjective genetic algorithm (MOGA)
Mohr's circle, 337, 337f
Molecular dynamics (MD), 1002
MOLP. *See* Multiobjective linear optimization (MOLP)
Moment–curvature curve (0AB) for an elastic-perfectly plastic sheet, 706, 706f
Monotonic deformation, 694–695
Monotonicity, 866
Monte Carlo method, 13
  simulation, 1152
MOO. *See* Multiobjective optimization (MOO)
MOSA. *See* Multiobjective simulated annealing (MOSA)
Most probable point (MPP) search algorithm, 555, 557b, 566f, 1151, 1152
Motion analysis, 11, 24
  computer-aided methods. *See* Computer-aided methods
  computer tools process, 427–428, 429f
  design parameterization, 396
  driving simulator, 454f
    Daimler-Benz Driving Simulator, 452–453
    human factors, 450–452
    NADS, 452–453, 455f
    Stewart platform, 408, 408f
  dynamic analysis, 395. *See also* Multibody dynamic analysis
    definition, 396
  GSTIFF, SI2_GSTIFF and WSTIFF, 432
  HMMWV, 394–395
    absorbed power equations, 449
    constraint functions, upper bound, 450, 451t
    design optimization of, 450, 451t
    design problem, 447–448
    front suspension, dynamic simulation, 448, 452f
    initial and optimal design, 450, 451t
    vehicle and suspension assembly, 445–446, 447f
    vehicle performance, 450, 453f
  kinematic analysis, 395. *See also* Computer-aided methods; Multibody kinematic analysis
    definition, 408–409
  motion model creation
    applied forces, 431
    3D contact constraint, 430
    degrees of freedom, 430–431
    flexible connectors, 431–432

ground parts, 428
harmonic function, 432
initial conditions, 432
motion drivers, 432
moving parts, 428
rail with path mates, 429–430, 433f
single-piston engine, 395, 395f
springs and dampers, 432
unconstrained rigid body, 428–429
particle motion
angular position, velocity and acceleration, 397b
energy method, 399
external and internal forces, 401
kinetic and potential energy, 399
Lagrange's equations, 399
moment equilibrium equation, 397b
multiparticle system, 401–404
Newton's law, 397
object sliding, parametric curve, 400, 400f
two-particle system, 401b
products and mechanical systems, 393, 394f
Pro/ENGINEER mechanism design real-time simulation, 393
recreational waterslides
flume section, 454, 457f
friction coefficient, 456
generalized friction forces, 455
geometric representation, 454, 457f
object path and unit vectors, friction forces, 454, 458f
safety problems, 454
second-order ordinary differential equations, 400
translation and rotation operations, 450t
results visualization, 433, 433f
rigid-body motion, 404–407
angular momentum, 404, 405f
definition, 404
kinetic energy, 407
mass moments of inertia, 405
potential energy, 407
rotational equation of motion, 404
translation equation of motion, 404
second-order differential equations, 400
simulation models, 10–11
mistakes in, 11
simulation software
Adams and DADS codes, 434
Adams/Car, 434–435
applications, 434
Formula SAE racecar model, 435–445, 437f. *See also* Formula SAE racecar model
IN-Motion, 434
Windows-based CarSim version, 435
single-piston engine, 459–460, 460f
exploded view, 395, 395f, 433f, 460f
unexploded view, 395, 395f
sliding block, 458–459, 460f
SolidWorks Motion static analysis, 432, 435f
Motion simulation models, 10–11, 11f
MPP search algorithm. *See* Most probable point (MPP) search algorithm
MSC/Abaqus software, 377
MSC Fatigue, 12, 506
MSC.Marc FEA program, 378
MSC/NASTRAN® software, 11–12, 978–979
MSC/Patran, 8–10, 377
M-series steels, 624
MTBF. *See* Mean time between failure (MTBF)
Multiattribute optimization. *See* Multiobjective optimization (MOO)
Multiattribute utility functions (MAU), 871–872, 1142. *See also* Utility theory
additive, 872–874
multiplicative, 874–875
Multibody dynamic analysis, 411–415
body-fixed reference frame, 412, 412f
computer-aided methods, 415–427
DAE, 426–427
Lagrange multipliers, 426–427
variational equation of motion, 425
velocity and acceleration equations, 428
equations of motion, 397
Euler equations, 412
interconnected rigid/flexible bodies, 411
nonlinear differential equations, 415
Multibody kinematic analysis
nonlinear algebraic equations, 415
relative velocity/graphical method, 409–411
slider-crank mechanism, 408–409, 409b, 420f
Stewart platform, 408, 408f
Multicriteria optimization. *See* Multiobjective optimization (MOO)
Multicriterion problem. *See* Multiobjective problem
Multiobjective genetic algorithm (MOGA), 1131, 1148–1150
advantage, 1131
MATLAB implementation, 1137–1141
NSGA II, 1133–1137
Pareto-based approaches, 1132–1133
Multiobjective linear optimization (MOLP), 1118–1119
lexicographic method, 1124b
NBI method, 1127b
pyramid problem solving, 1128b
weighted min-max method, 1122b
weighted-sum method, 1118b

Multiobjective optimization (MOO), 1106–1107, 1108b
advanced topics, 1151
manufacturing cost, 1158–1170
RBDO, 1151–1158
for structural performance, 1158–1170
criterion space, 1110–1111, 1110f
design space, 1110–1111, 1110f
GA, 1131–1141
Pareto optimality, 1111–1113
Pareto optimal set generation, 1113–1115
pyramid example, 1108f
reliability-based design optimization, 1107
software tools, 1145–1150
solution techniques, 1116
no articulation of preference, methods with, 1130–1131
objective function normalization, 1116–1117
posteriori articulation of preferences, methods with, 1125–1130
priori articulation of preferences, methods with, 1117–1125
terminologies, 1110
Multiobjective problem, 916
Multiobjective programming. *See* Multiobjective optimization (MOO)
Multiobjective simulated annealing (MOSA), 1148–1150
Multiphase problems, 11–12
Multiple failure modes
parallel system, 537f, 583–585, 592f
series system
failure element, 582, 582f
FORM approximation. *See* First-order reliability method (FORM)
frame structure, 582, 582f
Multiple variable functions, 920–923
Museum application using RP system, 762–763
Mutation, 967

N

NASGRO, 507
Nash equilibrium, 877
Nastran software, 340
National Advanced Driving Simulator (NADS), 408, 408f
NBI. *See* Normal boundary intersection (NBI)
nCode, 12
nCode DesignLife™, 505
NC part programming, 602–611
actions specified in, 606
approaches, 607f
5-axis CNC, 605
axis of motion, 603, 603f
CLU (control loop unit), 602
code verification, 608f
DPU (data processing unit), 602
machine control unit (MCU), 602
manual, 606–608
NC codes, 602
NC machines, basics, 602–605
profile milling, 606–608
PTP (point-to-point) or continuous path, 604
sequence of actions of NC machines, 606
servomotor (servo), 602
three-axis motion, 605
ways to classify machines, 604–605
workpiece-rotating machine, 604
NC programming
approaches, 607f
manual, 606–608
methods for creating, 606
NC sequence
5-axis mill with ball-nose cutter, 658
in Pro/MFG, 634f
profile milling, 622
Necessary condition, 919
Neighborhood of point, 923
NESSUS®. *See* Numerical evaluation of stochastic structures under stress (NESSUS®)
Neuber's rule, 481, 482f
Neutral file exchange, 297–298
IGES, 297
STEP (ISO 10303), 297–298
Newton iteration, 720
Newton's second law, 375
Niche techniques, 1133
NIMBUS system, 1145–1147
for pyramid example, 1146f
sample windows, 1147f
NLP. *See* Nonlinear programming (NLP)
Noncooperative game, 876–877
Nondominated solution, 1106
Nondominated sorting genetic algorithm (NSGA), 1133
fitness computation, 1132f
NSGA II, 1133
crowding-distance calculation, 1136f
niche technique, 1135–1136
nondominated sorting, 1134–1135
process, 1136–1137
Nondominated space, 1112
Nongeometric features, 140
Nongradient algorithms, 1003
Non-gradient-based approach, 910, 918, 964–970. *See also* Gradient-based approach
GA, 965–968
SA, 968–970

Nonlinear programming (NLP), 933–934
problem, 933–936, 933b
techniques, 917
Nonmanifold objects, 136–137, 136f
Non-normal distribution
correlated random variables, 573–575
independent random variables, 571
Nonuniform B-spline curves, 64–68, 65f, 67f–68f, 244
Normal anisotropy, 698
Normal boundary intersection (NBI), 1126–1130, 1126f, 1148–1150
NSF. *See* National Science Foundation (NSF)
NSGA. *See* Nondominated sorting genetic algorithm (NSGA)
Notification, product data management utility function, 282
Numerical control (NC) toolpath generation, 127–129, 130f
Numerical evaluation of stochastic structures under stress (NESSUS®), 13
Numerical modeling of metal forming processes, 707–709
explicit and implicit incremental formulations, 708–709
rigid geometric entities, 708
steps for conducting a representative FE simulation, 709, 710f
types of nonlinear analysis, 708
NUMISHEET'93 benchmark, 704f
NURB curves, 75–76, 75f
quadratic, representing conics with, 48f, 116–120
NURB surface, 83
NX, 7
Unigraphics, 629
viewer, 271

O

Objective function, 1166
normalization, 1116–1117
OEM. *See* Original equipment manufacturer (OEM)
Offset strain, 693
One-cut-point method, 966–967
Online consumer reports, 858
OP010, 668
OP030, 668
OpenForm, 725–726
Open-loop kinematic system, 210–212, 210f
adding parallel mating constraint between two planes, 213f
determining coordinate systems, 210f
link parameters for, 224t
z-axis and coordinate systems of joints, identification of, 221, 221f, 223f–224f
Optegra, 285
Optimality conditions, 918–930
design optimization, 923–924
Karush–Kuhn–Tucker conditions, 927–930
Lagrange multipliers, 924–927
multiple variables functions, 920–923
single variable functions, 919
Optimality criteria methods, 917
Optimization problem, 912, 1003
classification, 916–917
problem formulation, 913–915
cantilever beam example, 915f
design problems, 913
numerical simulations, 915
objective function, 914–915
performance measures, 914
physical modeling, 914
steps for, 913
traffic light, 914f
problem solutions, 915–916
solution techniques, 917–918, 918f
Optimization software, 978
CAD, 978–979
FEA, 979–980
special-purpose codes, 980
OPTIMUS®, 1150, 1150f
OptiStruct software, 980
Orderability, 866
Organ printing, 771
Original equipment manufacturers (OEMs), 268, 854–855, 854f
Orthogonality, 193

P

Palmgren-Miner linear damage hypothesis, 477
Pam-Stamp 2G, 726–727
Parabola, 117–118, 118f
Parachuting simulation, 227, 227f
Parameterized CAD product model, 7–8, 13–14
airplane engine model, 8f
analysis models, 8–10
challenges, 7–8
design change propagation, 9f
engine motion model, 11f
finite element meshes, 10f
motion simulation models, 10–11
single-piston airplane engine, 8
3-Parameters Barlat and the Krupskowsky strain-hardening model, 735–736
Parameters, product data type, 276
Parametric curves, 44–76, 44f–45f
B-spline curves, 64–74
continuities, 63
cubic curves, 56–62
NURB curves, 75–76, 75f
quadratic curves, 47–56, 48f–49f
straight line, 46–47, 47f

Parametric modeling, 131, 150–151, 244
Parametric product models
  airplane engine model, 8f
  in CAD, 7–11
  challenges in generating simulation, 7–8
  engineering characteristics of non-CAD parts and assemblies, 7
Parametric study design, 16–17, 17f. *See also* What-if study
Parametric surfaces, 76–83
  B-spline surface, 82–83, 83f
  representation, 77–82
Parasolid, 42, 138, 156, 161
Parent–child relationships between solid features, 150, 150f
Pareto
  criterion, 883
  optimal set generation, 1113–1115
  optimality, 1111–1113
  optimum, 1109
Pareto-based approaches, 1132–1133
  elitist strategy, 1133
  fitness assignments, 1132f
  niche techniques, 1133
  ranking, 1132–1133
Pareto optimization. *See* Multiobjective optimization (MOO)
Parts
  level, design parameterization at, 242–248
    guidelines, 245–248, 246t
    of HMMWV, 257–261, 259f
    profile in sketch, 242–244
    of single-piston airplane engine, 256
    solid features, 244–245
  positioning, 184
  Pro/ENGINEER, importing, 11–12, 292–294
    converted gear housing model, 293f
    converted solid model and features, 293f
    Converter dialog box, 293f
    gear housing part in, 293f
    translation report, 293f
Part programming
  basic concepts, 606–608
  CAD/CAM approach, 611
  computer-assisted, 608–611
  for turning simulations, 621. *See also* NC part programming
PartsList, 288
Pattern features, 141, 142f
PATRAN, 278, 308–309
PDM. *See* Product data management (PDM)
PDXViewer, 288
Peer-to-peer file sharing, 270
Penalty function, 962–964
Penalty method, 962–964, 963b
Performance analysis, product, 11–13
  at component level, 26t
  fatigue and fracture analysis, 12
  motion analysis, 11
  reliability evaluations, 12–13
  structural analysis, 11–12
Performance measure approach (PMA), 565, 1152
Performance measures, 1003
  compliance, 1050
  pitfalls effect, 1081–1082, 1082f
  sensitivity analysis for, 1005
  structural, 1012
  variation, 1051
Personal rapid prototyping (RP), 771
Perturbed curve, 1060
Physical prototyping, 19–22. *See also* Virtual prototyping (VP)
  computer numerical control machining, 21–22
  rapid prototyping, 19–21
Physics-based simulation technology, 6
Pick-and-place, 131, 131f, 139, 141
Piecewise constant function, 1012
Pin joint, 178, 180f
PISA framework, 1145
Plain milling machine vise, 624
Planar curves, 46
Planar joint, 179–181, 180f
  mapping mating constraints to, 201f, 202
  origin and z-axis of, 219, 220f
2-D Planar structures, 1058–1060, 1068–1070
Plane stress sheet deformation, modes of, 696–697
Plane strain bending condition, 704–705
Plane stress, 693–697
Plans, solid modeling, 151–153
  build plan, 157–160, 158f–160f
Plastic deformation, 692, 700
Plastic work done in deforming a unit cube, 695
PMA. *See* Performance measure approach (PMA)
Polar coordinate system, circle in, 45, 45f, 75
Polydimensiloxane (PDMS), 766
Polymerization, 750
16-point format, 78–79, 79f
Poisson's ratio, 375
Population, 965
  drift, 1133
Posteriori articulation of preferences, methods with, 1125–1130
Postoptimum study, 986
Powder-based RP systems, 752–754
Power law, 693

Practical engineering problems, 970–978
  commercial CAD/CAE tools, 970
  interactive design process, 975–978
  tool integration for design optimization, 971–974
Preferences
  methods with no articulation of, 1130–1131
  methods with posteriori articulation of, 1125–1130
  methods with priori articulation of, 1117–1125
Pressure vessel
  design, 894–895
  example, 1144, 1144f
Pre-strain, 693
Primitives, 130, 131f, 133, 133f
Principal vectors, 197
Priori articulation of preferences, methods with, 1117–1125
Prismatic joint, 178, 182f, 183, 200, 206
  mapping mating constraints to, 201f
  origin and z-axis of, 219, 220f
Probability density function (PDF), 528, 528f, 538, 538f
Probability of failure, 13
Pro/CASTING, 13
Product data exchange, 289–301
  data exchange options, 291–292
  direct model translations, 292–296
    data exchange between CAD and CAE/CAM, 296
    importing Pro/ENGINEER assembly to SolidWorks, 296
    importing Pro/ENGINEER parts to SolidWorks, 292–294
    importing SolidWorks assembly to Pro/ENGINEER, 296
    importing SolidWorks parts to Pro/ENGINEER, 295
  neutral file exchange, 297–298
    IGES, 297
    STEP (ISO 10303), 297–298
  solid feature recognition, 299–301
  third-party translators, 298
    Proficiency, 298
    TransMagic, 298
Product data management (PDM), 4, 266
  case studies, 301–310
    SolidWorks Workgroup PDM, 302–303
    integrated testbed using Windchill, 303–305
    tool integration for e-Design, 305–310
  file management, 269–274
    ad-hoc methods, 270–273
    PDM approach, 273–274
  fundamentals, 274–285
    impact to industry, 284–285
    process data model, 279–280
    product data model, 275–279
    user functions, 281–282
    utility functions, 282–283
  IGES file structure and data format, 311–316
    directory entry section, 312
    global section, 312
    parameter data section, 312–315
    start section, 312
    terminate section, 315–316
  product data exchange. *See* Product data exchange
  step data structure and applications protocols, 316–317
  systems, 285
    approach, 273–274
    offered by CAD vendors, 286–288
    offered by non-CAD vendors, 288–289
Product development cycle, 2
Product life cycle design, 3
Product life cycle management (PLM), 266–267
ProductCenter, 288
Product performance evaluation, 11–13
  fatigue and fracture analysis, 12
  motion simulations, 11
  product reliability evaluations, 12–13
  structural analysis, 11–12
Product reliability evaluations, 12–13
  challenges, 13
Product structural analysis, 8–10
Product virtual manufacturing, 13
Pro/ENGINEER, 7, 10, 13, 27, 126, 142, 153, 155, 161, 170, 173–178, 181, 183–184, 210, 212, 234, 242, 245, 248, 250–251, 278, 286t, 629, 747–748, 767
  importing parts and assemblies to SolidWorks, 8–10
    assembly, importing, 11, 294–295
    gear train assembly in, 28, 28f
    parts, importing, 11, 295
  mating constraints in, 174t
  sketch relations in, 163t
  slider-crank assembly in, 250–251, 252f
  solid modeling with, 42
Problem formulation design, 15–16
Procedural parametric modeling. *See* Parametric modeling
Product data
  categories, 275
  model, 275–279
Profile milling, 606–608, 607f, 615f
  cutting parameters, 614
  hole-making, 600, 616f
  NC sequence, 614–615, 621
  smooth inner pocket boundary, 615
  sequence, 611
  step depth, 614
  step over, 615
  step sideways, 615
  volume milling, 618
Profiles, sketch, 142–149
Pro/Intralink, 285
Program management, 282

Project management, 282
ProMax-One™, 823
Pro/MECHANICA, 7–12, 979, 995
Pro/MFG, 13, 22, 600, 611, 617, 634
  5-axis mill with ball-nose cutter, 658
  for CNC simulation, 611
  gouge-checking capability, 679
  module, 13
  scallop height calculation, 660
Pro/MOLD, 13, 632
Pro/PDM, 285
Proportional deformation, 694–695
Prospector®, 823
Pro/SHEETMETAL, 13
Pro/TOOLKIT function, 1163
Prototypes, fabrication of, 2–3
Protrusion, 84, 84f, 131, 131f, 141, 141f
Pseudo-objective function, 963
PTP (point-to-point) machine, 604
Push-off factor, 959–961
p-version finite element analysis, 10, 10f
Pro/WELDING, 13

Q

Quadratic curves, 47–56, 48f
  Bézier curve, 48f, 62
  B-spline curve, 65f, 71–74
  spline curve, 48–50, 48f–49f
  two points and a vector, 50–52
Quadratic programming (QP), 17–18
Quality functional deployment (QFD), 15
Quasi-Newton method, 944
QuickCast, 761

R

Rail and carriage subsystems, 172, 172f
Ranking, 1132–1133
Rapid manufacturing, 745–746
Rapid prototyping (RP), 2, 6, 19–21, 26, 129. *See also* Virtual prototyping (VP)
  advanced, 754–759
  art applications, 761–763
  for airplane engine, 8f, 23–26
  bioengineering application, 768–771
  CNC machining, 21–22, 22f
  commercial systems, 19–21, 20f–21f
  crosshatch pattern of typical cut-out layer, 22f
  custom prosthesis and implantation, 768–769
  dental application, 767–768
  design applications, 760–761
  education and research application, 766
  electron beam melting (EBM), 755–756
  FDM, 20
  in film production, 763
  general process, 746–747
  in Jurassic Park III, 746f
  laser engineered net shaping (LENS), 757
  layered manufacturing, 20f
  liquid-based RP systems, 750–751
  manufacturing applications, 761
  medical applications, 763–768
  micro-manufacturing, 757–759
  model conversion, 780
  model fabrication, 780–781
  model modification, 777–780
  museum application, 762–763
  organ printing, 771
  personal, 771
  physical prototypes, 21
  powder-based RP systems, 752–754
  racecar model, 775–781
  scaffolds in tissue engineering, 769–770
  SFF technology, 19, 20f
  STL engine case models, 20, 21f
  single-piston engine assembly, 773–774
  solid-based RP systems, 751–752
  Solidica, 754–755
  STL engine case models, 20, 21f
  in surgical operations, 763–766
  technology touches, 771–772
  two-layer geodesic sphere, 772–773
Rectangle scaling, 102, 102f
  translation, 103–104, 104f
Reduced global stiffness matrix, 1011
Regular point, 925
Relations, sketch, 142–144, 143f, 162, 163t
Reliability analysis
  failure mode, 526
  failure probability. *See* Failure probability
  FORM, 553–560. *See also* First-order reliability method (FORM)
  general purpose tools
    ANSYS PDS, 589
    FE-RA software, 588
    FERUM, 589
    NESSUS, 588
    physical parameter uncertainity, 588
  HMMWV, 525
    CDF graphs, 584f, 590–591
    crack initiation life prediction, 589–590, 590t
    fatigue life contour, 589, 590f
    FE model and geometric parameters, 557f, 589–590
    rigid-body dynamic simulation, 588

irreducible uncertainties, 525
limit state function, 548, 549f
Monte Carlo simulation, 549–552, 581
importance sampling method, 575–578, 576f
MPP search, 581
multiple failure modes. *See* Multiple failure modes random variable transformation, 568
non-normal distribution, correlated random variables, 573–575
non-normal distribution, independent random variables, 571–573
normal distribution, correlated random variables, 568–571
probability and distribution functions, 570b
reducible uncertainties, 525
response surface method, 579–580
safety factor approach, 524–525
SORM, 524, 547–548, 566–568, 566f
statistics and probablistic theory. *See* Statistics and probablistic theory
Reliability-based design optimization (RBDO), 565, 1151. *See also* Multiobjective optimization (MOO)
design variable, 1158t
failure probability, 1152
gradient-based, 1153f
problem formulation, 1152–1155
for tracked-vehicle roadarm, 1155–1158
Reliability-based design sensitivity analysis, 1153–1155
Reliability evaluations, product, 12–13
Reliability index approach (RIA), 560–563, 561f, 1152
Removable media, 270
Renault–UNISURF 1971, 160
Representation, parametric, 77–82
Bézier surface, 81–82
bicubic surface patch, 77–78, 79f
Coons patch, 79–81, 79f
16-point format, 78–79, 79f
Reproducing kernel particle method (RKPM)
B-spline kernel function, 357, 357f
displacement interpolations, 357f
engine mount, 358, 358f
stiffness matrix, 358
Reproduction process, 966
Response surface method, 579–580
Reverse engineering, 112, 114f
Revolute joint, 178, 180f, 202–204, 206
mapping mating constraints to, 201f, 202
origin and z-axis of, 219–222, 220f, 222f
Revolved surfaces, 92–95, 93f
Rhinoceros, 161–162
RIA. *See* Reliability index approach (RIA)
Risk, decision under, 859
car-buying example, 860–861, 861f
chance node, 860
cost of options and events, 860t
decision node, 860
expected payoff, 859–860
historical data, 859–860
Roadarm manufacturing process, 632f
Roadarm solid model reconstruction, 1096f
Roadwheel sizing optimization, 981
analysis model, 981–983
design optimization, 986
design sensitivity
matrix, 983t
results and display, 983–984
finite element model, 983f
geometric modeling and design parameterization, 981, 982f
performance measures, 983
postoptimum study, 986
tracked vehicle roadwheel, 982f
trade-off determination, 985, 985t
what-if study, 984–985
Robotics, 200, 202, 204–205, 211
Romulus, 161
Rosenblatt transformation, 573
Rotation transformations, 104–105, 105f, 188–189, 193
DOF analysis, 197t
Rough-Restmill, 639
Ruled surfaces, 87–89, 87f
toolpath generation of, 657–676
3-axis mill with ball-nose cutter, 671–674
3-axis mill with flat-end cutter, 668–670
4-axis mill with flat-end cutter, 674–676
5-axis mill with ball-nose cutter, 658–667
CC and CL curves, 669–670
CL data, 662–666
flat-end cutter toolpath generation, 669–670
parametric surface and CL data, 662–666
scallop height, 659–661
RP. *See* Rapid prototyping (RP)

S

Safe Technology Ltd., 506
Safety margin, 548
SA. *See* Simulated annealing (SA)
SAU. *See* Single attribute utility (SAU)
Scaffolds in tissue engineering, 769–770
Scalarization method. *See* Weighted-sum method
Scallop height calculation, 653, 654f–655f
5-axis mill with ball-nose cutter, 659–660

Scaling transformations, 102–103, 102f
Schemes, solid modeling, 132–138
  boundary representation, 135–138, 135f
  constructive solid geometry, 132–135
Screw joint, 180f, 181
SDRC, 157t, 161, 285
Search methods, 937–939
  design variables, 939
  equal interval, 937–938, 937f
  golden section, 938, 938f
Secant method, 948–949
Second-order reliability method (SORM), 13, 524, 547–548, 566–568, 566f, 1152
Second-order Taylor series expansion, 566
SEER for Manufacturing (or SEER-DFM), 832–837
Selective laser sintering (SLS), 752
Self-adjoint response, 1051–1052
Self-centering 3-jaw chuck, 623
Semiautomatic mesh generation
  bi-cubic parametric patch, 370, 370f
  quad- and hex-elements, 369–370
  three-dimensional turbine blade, 371, 371f
  two-dimensional connecting rod, 370, 370f
  SEQA, 476
Sensitivity analysis design, 16, 25–26, 1002
  methods, 1012
    analytical derivative method, 1014–1015
    continuum approach, 1013, 1024–1029
    discrete approach, 1017–1024
    finite difference method, 1015–1017
    sensitivity formulation, 1013
  for performance measures, 1005
  reliability-based design, 1153–1155
Sensitivity formulation, 1013
Sequence using a CNC machine, 627–629
  key steps, 627
  loading of cutters, 628
  X position of part, 628
  Z-axis, 628–629, 628f
Sequential games, 880
  backward induction, 881
  game tree, 877, 880f
  Nash equilibrium, 881
  solution of, 881
  Stackelberg duopoly model, 882
  strategy profile, 881
Sequential linear programming (SLP), 918, 949–950, 952–956
  linearized objective function, 952
  mathematical notations, 952–953
  objective and constraint functions, 953–956
  optimization problem, 953b
Sequential quadratic programming (SQP), 918, 1168–1169
  algorithm, 956–957
Servomotor (servo), 602
SFF. *See* Solid freeform fabrication (SFF)
Shape optimization, 111, 111f
  design optimization, 1097–1099
  engine connecting rod, 986–987, 990f
    design optimization, 990
    design parameterization and problem definition, 989–990
    geometric and finite element models, 988–989
Shared network folders, 270–271
Shape sensitivity analysis, 1053. *See also* Design sensitivity analysis (DSA); Topology optimization
  cantilever beam example, 1054
    deformed cantilever beam, 1056f
    discrete-analytical approach, 1055
    displacement sensitivity, 1056
    midpoint movement with design change, 1057f
    shape design variable, 1055
    shape sensitivity coefficients, 1057
  using continuum approach, 1087–1088
    cantilever beam, 1088–1090, 1089b
    material derivative, 1087–1088
  design velocity field computation, 1065
    boundary velocity computation, 1068–1075
    design velocity fivel, 1066–1068
    domain velocity computation, 1075–1080
  domain, 1053–1054
  using finite difference method, 1080–1082
  material derivatives, 1082, 1083f
    deformed cantilever beam, 1084f, 1084b
    design velocity field, 1086
    of displacement, 1086
    energy bilinear form, 1082
    FEA solution, 1084
    gradient operator, 1083
    load linear form, 1082
  shape design parameterization, 1058
    2-D planar structures, 1058–1060
    3-D solid structures, 1061–1063
Sheet hydroforming processes, 691–692, 691f
  advantages, 692
  disadvantages, 692
  maximum pressure for, 691–692
Sheet metal forming simulation
  addendum design, 716–718
  advantages, 687
  aluminum 2024 sheet, principal stresses and sheet thickness, 696b
  anisotropic yield criteria, 700–701
  binder surface (or blank holder surface), 716
  blank fitting and blank nesting, 713–714

case studies, 730–739
computational aspect, 686–687
core panel, problem with, 731
die design, 714–718
draw forming (or deep draw), 689–690
drawbead design, 718
forming limit diagram (FLD), 701–704
fundamentals, 689–709
general plane stress sheet processes, 692–698
incremental forming analysis, 719–721
isotropic yield criteria, 699
material anisotropy, 697–698
monotonic and proportional deformation, calculating, 694–695
numerical modeling of metal forming processes, 707–709
one-step simulation, 711–714
part preparation, 715
process planning and tooling design, 709–723
sheet hydroforming (fluid forming), 691–692
springback analysis, 704–707
springback analysis and die compensation, 721–723
stress–strain curve, 692–693
stretch forming, 690–691
wheel fairing, 734–739
yield criteria, 698–701
Sheet metal stamping cost model, 816–819
approximating material cost, 816–817
assembly cost model, 819
material costs, 816–817
metal properties and typical costs, 817t
processing cost, 818
tooling costs, 819
two-step stamping process, 816f
Shock absorber operation distance, 30f
Side constraints, 913
Sidewall curl, 704
Siemens UGS NX, 286t
Siemens PLM Software, 272
Simple airplane engine, 23
component-level design, 25
changes in design variables, 26t
changes in performance measures, 26t
design trade-off method, 25–26
system-level design, 23–24
changes in design variables, 24t
engine assembly with design variables, 23f
Simplex method, 932
Simulated annealing (SA), 965, 968
algorithm, 969
cooling process, 969
solution process, 970
"temperature" parameter, 969
Simulation-based process planning and tooling design process, 709–723
addendum design, 716–718
binder surface (or blank holder surface) design, 716
blank fitting and blank nesting, 713–714
die design, 714–718
drawbead design, 718
flowchart, 711f
incremental forming analysis, 719–721
one-step simulation for formability, 711–714
part preparation, 715
springback analysis and die compensation, 721–723
Simulation models
dynamic, 29f
files, 275–276
motion, 10–11, 11f
Single attribute utility (SAU), 871–872
Single-piston engine, 227–228, 228f, 253–257, 254f, 773–774
assembly mating constraints defined for, 228f
design parameterization of, 253, 254f
at assembly level of, 256–257
design variables design for DIs, 255f
excessive information, 254f
at part level, 255–256, 255f
Single variable functions, 919
Single-criterion problem. *See* Single-objective problem
Single-objective optimization, 1107. *See also* Multiobjective optimization (MOO)
problem, 1159
Single-objective problem, 916
Sinterstation®, 19–20, 20f
Six Sigma, 3
Sizing and material designs, 1029. *See also* Design sensitivity analysis (DSA)
continuum-discrete method, 1029–1030
numerical implementation, 1052–1053
static problems, 1037–1038
adjoint variable method, 1046–1050
bending stress sensitivity, 1044–1046
cantilever beam example, 1038f
direct differentiation method, 1043–1044
energy equation, 1038–1039
finite element discretization, 1040–1042
variations, 1033–1037
definition, 1033–1036
of energy bilinear and load linear forms, 1036–1037
virtual work principle, 1030
differential equation, 1030–1031
Sketch profiles, 142–149, 242–244
sketch relations, 142–144, 143f, 162, 163t
variational modeling, 144–149
Skinning. *See* Surface skinning

SLA®7000, 19–20, 21f
SLAViper, 750
Slicing software, 20, 747
Slider joint, 177f, 178
Slider-crank mechanism, 177f, 178, 181, 973f
  as closed-loop system, 214–218, 215f
    link parameters for, 224t
    z-axis and coordinate systems of joints, identification of, 221, 222f
  crank rotated in, 200f
  design changes in, 199f
  design parameterization, 234, 239, 239f–240f
    assembly in Pro/ENGINEER, 250–251, 251f
    assembly in SolidWorks, 252–253, 252f–253f
    constructive solid geometry tree, 249f
    design intent capture, 248f
    feature creation steps of crankshaft, 247f
    variational equations for sketch profile, 247f
  kinematic characteristics of, 180f
  kinematic model of, 182f
  locations of datum points and axes, 184f
  mating constraints defined for, 179f
  as open-loop system, 210–212, 210f, 213f
    link parameters for, 224t
    z-axis and coordinate systems of joints, identification of, 213f, 221, 222f–223f
  simulation model joints, 183t
  using SolidWorks, 200–202, 227–228
SLP. *See* Sequential linear programming (SLP)
Smith, Watson, and Topper (SWT) model, 483
SMLib, 156
Smooth particle hydrodynamics (SPH) method, 356
SofTech ProductCenter PLM, 288
Software as a Service (SaaS), 268
Software implementation, 1163–1164, 1164f
Software tools, 1145
  academic codes, 1145–1147
  commercial tools, 1147–1150
Solid-based RP systems, 751–752
Solid Edge, 161
Solid features, recognition, 299–301
Solid freeform fabrication (SFF) technology, 19, 20f. *See also* Rapid prototyping (RP)
Solidica, 754–755
Solid modeling, 126, 130–132, 131f, 242
  basics of, 127–138
  build plan, 157–160, 158f–160f
  commercial CAD systems, 160–162
  feature-based parametric solid modeling, 139–157
3D solid models to surface (shell) or curve (beam) models, 10
3-D Solid structures
  CAD-generated surfaces, 1063–1065, 1072–1073
  extrusion of sketch profile, 1065f
  freeform surfaces, 1061–1063, 1071
  parametric cylindrical surface, 1066f
SolidWorks, 7, 10, 111, 126, 141–143, 153, 155, 162, 170, 173–175, 180f, 184, 227–228, 234, 242, 244–245, 248, 252, 288, 292–295, 302–303
  advanced mates in, 176t, 177f
  importing parts and assemblies to Pro/ENGINEER parts, 295
  mating constraints in, 175t
  Rebuild Errors window, 152f
  sketch relations in, 163t
  slider-crank assembly in, 252–253
    change of length design variables, 253f
    mating constraints, 252f
  standard mates in, 176t. *See also* Pro/ENGINEER
SolidWorks Enterprise product data management, 288, 302
SolidWorks Motion, 10
SolidWorks Simulation, 377
  one-dimensional beam model, 381–382, 382f
  simple beam model, 381, 382f
  thin-walled tubing model, 382–383, 383f
  tube surface model, 383–384, 383f
  user interface, 434
SolidWorks Workgroup product data management, 302–303
Solution techniques, 917–918, 918f
SORM. *See* Second-order reliability method (SORM)
SpaceClaim Engineer, 162
Spatial curves, 46
Spherical joint, 180f, 181, 209
  mapping mating constraints to, 201f, 202
  origin and z-axis of, 220f, 221
Spindle speed, 625–626
Spline curve
  B-spline curves, 64–74
  cubic, 56–62, 56f
  quadratic, 47–56, 48f–49f
Springback analysis, 704–707, 721–723
  defined, 723
  flowchart of simulation-springback-compensation procedure, 724f
  purpose of, 723
  shape of a sample part before and after, 723f
  springback compensation, 721–723
SQP. *See* Sequential quadratic programming (SQP)
Stackelberg duopoly model, 882
Stamping direction, 715
Standard for Exchange of Product (STEP) model data, 297–298, 372
Standard normal density function, 1154
State of nature, 854

States of knowledge, 858
Static problems, 1037–1038
  adjoint variable method
    of continuum-analytical approach, 1050–1052
    of continuum-discrete approach, 1046–1050
  bending stress sensitivity, 1044–1046
  cantilever beam example, 1038f
  direct differentiation method, 1043–1044
  energy equation, 1038–1039
  finite element discretization, 1040–1042
Static workflows, 279
Stationary point, 919, 919f
Statistics/probablistic theory
  continuous random variable, 538
  discrete random variable, 538
  distribution functions, 538–539, 538f
  extreme value distributions, 546–547, 546f
  joint probability density function, 539–543
    correlated random variables, 540, 540f
    covariance matrix, 541
  lognormal distribution function, 545, 546f
  mean value and standard deviation, 539
  normal distribution function, 544
  probability rules
    Bayes' theorem, 537
    conditional probability, 536
    individual events, 535
    sample space and event, definition, 535
    total probability theorem, 536
Steepest decent method, 940–942
  minimum point of objective function, 941b
  orthogonal steepest decent path, 941f
Step data structure and applications protocols, 316–317
STEP model data. *See* Standard for the exchange of product (STEP) model data
Step over, 615
Stereolithography (STL) format, 20, 112, 129, 130f
  models, 20
    engine case models, 21f
StereoLithography Apparatus (SLA), 750
STL format. *See* Stereolithographic (STL) format
Stochastic programming problem, 917
Stockholm Metro's Car 2000, 3
Straight line, 46–47, 47f
Strain-hardening (or work hardening), 693
Strategy profile, 877–878
Strength coefficient, 693
STRESSCHECK p-version FEA code, 986–987
Stress intensity factor, 484
Stress-life approach
  complex multiaxial stress, 476–477
  endurance limit, 470
  in-phase bending and torsion, 475–476
  Miner's rule, 477
  nonfully reversed cyclic loads, 472–474
  stress-strain (S-N) diagram, 468
    endurance limit, 470
    fatigue strength, 470
    high-cycle fatigue, 470
    log-log diagram, 470, 470f
    maximum stress, 471
    stress error, 472
Stress–strain ratios, 694–695
Stress–strain relationship in elastic–plastic state, 692
Stretch forming, 690–691
Stroke, defined, 23–24
Strong local minimum function, 923
Structural analysis, 11–12
  default in.-lbm-sec unit system, 389
  energy method
    Castigliano's theorem, 330–331, 332b
    displacement calculation, 331
    strain energy, 331, 331f
  failure criteria
    flowchart for, 339f
    maximum shear stress theory, 336–337
    Mohr's circle, stress element, 337, 337f
    Tresca's hexagon, 337–338, 338f
    uniaxial stress load, 337, 337f
  FEMs, 326. *See also* Finite element methods/analysis (FEMs/FEA)
  linear elasticity
    infinitesimal strains/"small" deformations, 333
    stress components, 333f, 334, 335b
    stress function, 334, 335b
  linear isotropic infinitesimal elasticity, 333–336
  material strength, 336–337
  safety factor approach, 338
  structural components, 327, 327f
  uncertainty, 339
  variability, 339
Substitutability, 866
Sufficient condition, 919, 921–922
Suh's axiomatic design, 848
Super-IH micro-stereolithography, 759f
Supply Chain, ENOVIA Smarteam, 287
Surface-finish criteria, 649–650
Surface finish-scallop machining, 639–640, 640f
Surface gouging, 679f
Surface milling for tolerance, 679f
Surface models, 129–130, 129f–130f
Surface rough restmill machining, 639–640, 640f

Surfaces
  CAD-generated, 84–99
  parametric, 76–83
Surface skinning, 43–44, 108–111
  engineering applications, 111–112
Surface to surface intersection, 134, 134f
SurfCAM, 630
Sweep features. *See* Protrusion
Sweep surfaces, 96–99, 96f
Swift's law, 693
Swivel base, 624
SWT model. *See* Smith, Watson, and Topper (SWT) model
Synchronous interactions, 303
System-level design, for airplane engine, 23–24
  brake mean effective pressure, 23
  connecting rod, 14f
    dynamic load applied to, 24f
  design variables at system level, 23–24, 24t
  engine assembly, 23f
  rotational speed of crankshaft, 23
  variables, 24t
System motion simulations, 11

T

Tangent vectors, 665
Taylor's cutting speed equation, 809–810, 809t
Taylor series, 19, 75, 1006, 1034
TeamCenter, 286t, 287–288
Telepresence, 225–226
Terminal node, 881
Thermal analysis, 12
Thickened features, 141
Thinkernel, 156
Third-party translators, 298
  Proficiency, 298
  TransMagic, 298
Three-axis motion, 605
Three-step design process, 975
Time
  cost/ECR *vs*., 3f, 4
  functions, 1030–1031
  product knowledge *vs.*, 5f
Titanium carbide (TiC), 624–625
Tool integration, 4, 13–14, 283
  for e-Design, 305–310
    CAD and base definition, 307
    design collaboration, 310
    design process management, 309–310
    disciplines and views, 307–308
    engineering tool wrappers, 308–309
  design optimization, 971–974
    CAD-based mechanism optimization, 971, 972f
    change of length dimensions, 974f
    commercial codes, 971
    complete motion model, 972
    DSA, 972–973
    slider-crank mechanism, 973–974, 973f, 975t, 976f
    solid model, 971–972
Toolpath generation
  CC-based (cutter contact) method, 648–649
  CL-based method, 649–650
  contour surface milling, 648–649
  cylindrical surface of Bézier curve, 677–679
  drive surface method, 649
  guide plane method, 649
  inclined flat surface, 650–657
  objective, 648–649
  parametric method, 649–650
  ruled surface, 657–676
Tool wrappers, 14, 308–309
Toolpath files, 275–276
Tooth modeling, 112, 113f
Topology optimization, 111, 111f, 1004–1005, 1090, 1095. *See also* Design sensitivity analysis (DSA); Shape sensitivity analysis
  numerical instabilities in, 1093f
  problem formulation, 1091–1093
  and shape optimization, 1094–1095
    boundary smoothing, 1095–1096
    design velocity field computation, 1096–1097
    roadarm solid model reconstruction, 1096f
    shape design optimization, 1097–1099
    shape parameterization, 1096–1097, 1097f
    tracked vehicle roadarm, 1095
    von Mises stress, 1098f
  two-dimensional cantilever beam example, 1093–1094, 1094f
  types, 1091f
Tosca Structure, 980
Tracked vehicle roadarm surface model, 111, 111f, 141, 1095, 1155–1158
  deterministic design optimization, 1155–1156
  probabilistic fatigue life predictions, 1156
  reliability-based design, 1158
Trade-off analysis, design, 17–18, 17f–18f
  for airplane engine, 23–26
  for HMMWV, 32–35, 33f–34f
Transformation matrix, 185–197, 185f
  coincident, 186, 187f
  computation of, 188–197, 188f, 190f, 197f
  concentric, 186–188, 187f
Transformations, geometric, 100–108
  homogeneous coordinates, 100–102, 101f
Transitivity, 866

Translations, geometry, 103–104, 104f, 188–197
DOF analysis, 198t. *See also* Direct model translations
Translational joint, 178, 180f
TransMagic, 298
Tresca's hexagon, 337–338, 338f
Tresca yield criterion, 699
True strain, 692
True stress, 692
True stress–strain curve, 692
T-series steels, 624
T-slots, 624
Tungsten carbide (WC), 624–625
Turning simulations, 621
Tuning parameters, 967–968
Two-cut-point method, 966–967
Two-up nesting, 714

U

Uncertainty, 849, 853
built-in, 853
decision under, 861
Bayes' theorem, 861–862
car-buying example, 862, 863f
using posterior probability, 863, 863f
probability data, 863
designer's preference, 849
UNCL01.P11, 621
Unconstrained problems, 916
Uncoupled design intent, 236, 236f, 238, 239f, 248
Uniaxial tensile test, 692, 697–698
Unidirectional parametric modeling. *See* Parametric modeling
Uniform B-spline curves, 69–71
Uniform global scaling, 103
Union of primitives, 132, 133f
Uni-Solid, 161
Universal joint, 172, 180f
UPG2, 156
USB drive, 270
Utility theory, 864, 1141–1143. *See also* Decision theory; Game theory
assumptions, 864
compound lottery, 865f
equivalent simple lottery, 865f
lottery diagram, 864f
attitude toward risk, 867
average payoff, 867
decision tree, 868, 868f
expected utility hypothesis, 867
normalized parameter and utilities, 872t
St. Petersburg paradox, 867
utility function, 869, 869f
as design tool, 885, 894
cantilever beam example, 885f
constrained problem, 890–894, 893t
MAU function and optimum solution, 889f, 891f
nonconstrained design problems, 891t
SAU function for stress constraint, 892f
unconstrained problem, 886–890
utility functions, 887f
utility axioms, 865–866
utility functions, 866–867
concave, 869
construction, 870–871
mathematical model, 871f
risk-averse, risk-neutral, and risk-prone, 870f
Utopia point, 1113

V

Variational modeling, 142, 144–149, 151–153
Variables. *See* Design variables (DVs)
VDA, 291
Vector-by-vector micro-stereolithography systems, 758
Vector-evaluated genetic algorithm (VEGA), 1131–1132
Vector optimization. *See* Multiobjective optimization
Venkataraman's FR algorithm, 299
Vericut, 611
Vertical stretch forming process, 691f
view3dscene, 278
Virtual adjoint response, 1027
Virtual manufacturing, 13, 14f, 25
cover die machining, 22f
Virtual environment, 3
Virtual machining (VM), 14f, 1168, 1169f
model, 1159–1162
process, 14f
simulations, 611–621
address codes, 644
advanced, 616–621
of basic machining operations, 612–616, 613f
block with a sculpture surface, 617
case study and tutorial examples, 632–640
contour surface milling, 616
cutter approaches, 614
designing and creating a customized name plate, 636
fast feedrate, 618
G- and M-codes, 611
hole-making, 612
local milling, 616–617
M-codes and G-codes, 611
NC sequence, 614–615
pocket boundary, 615
profile milling sequence, 610f, 611

Virtual machining (VM) (*Continued*)
step over and step depth, 615
using CAD/CAM tools, 611
volume milling, 616–617
Virtual manufacturing, 25
CL (cutter location) data format, 600
die casting, 600
mold machine simulation, 601–602
NC part programming, 602–611
NC sequences, 601–602
simulation-based technology, 600
tools, 13
Virtual prototyping (VP), 2–3, 7–19, 15f. *See also* Physical prototyping
chip formation, 13
design decision making, 15–19
design problem formulation, 15–16
design trade-off analysis, 17–18
DSA, 16
parametric study, 16–17, 17f
what-if study, 19
design problem formulation, 15–16
design trade-off analysis, 17–18, 18f
DSA, 16
hierarchical product models, 15f
parameterized CAD product model, 7–11
airplane engine model, 7–8, 8f. *See also* Airplane engine
analysis models, 8–10
challenges, 7–8
design change propagation, 8, 9f
engine motion model, 10, 11f
finite element mesh, 8–10, 10f
motion simulation models, 10–11
non-CAD data, 7
product performance analysis, 11–13
fatigue and fracture analysis, 12
motion analysis, 11
physics-based simulation, 6
product reliability evaluations, 12–13
reliability evaluations, 12–13
structural analysis, 11–12
product performance evaluation, 11–13
product virtual manufacturing, 13
QFD, 15
Taylor series expansion, 19
tool integration, 13–14, 15f. *See also* High-mobility multipurpose wheeled vehicle (HMMWV) design
virtual machining process, 13, 14f
Virtual reality, 225–227, 226f–227f
Virtual Reality Modeling Language (VRML) Virtusphere, 227, 227f, 278
VM. *See* Virtual machining (VM)
Volume milling, 618
and drilling, 633
for the center hole, 635
for the middle hole, 636f
in Pro/MFG, 639
von Mises stress, 1098f
von Mises yield condition. *See* Isotropic von Mises yield criterion
VP. *See* Virtual prototyping (VP)
VSO BIOACT, 749–750

W

W-axis, 604
Weak local minimum function, 923
Weak Pareto optimal, 1112, 1112f
Web-based costing tools, 823
Web folders, 270–271
Weibull distribution, 547, 547f
Weighted-exponential sum method. *See* Weighted-sum method
Weighted min-max method, 1122–1123, 1123f
Weighted-sum method, 1117–1122
pyramid problem solving, 1120b
Weighted Tchebycheff method. *See* Weighted min-max method
What-if study, 19, 25–26, 32, 976–977
Wheel fairing, 734–739
Windchill, 286t, 287
Windows-based CarSim version, 435
Windows Messenger, 270
2D wireframe, 612–613
Wireframe models, 127–128, 128f
Workflow and process management, 282
Workgroup product data management, 302–303
World coordinate system (WCS), 173–174, 178, 222

X

XFEM. *See* Extended finite element method (XFEM)
XYZ system, 603, 603f
W-axis, 604
on workpiece-rotating machine, 604
Y-axis, 604
Z-axis, 603

Y

Yield criteria, 698–701
Yield function, 698

Z

ZCorp's 3D Printing (3DP), 752