



Introduction to Optimum Design

Second Edition



Jasbir S. Arora

Introduction to Optimum Design

Introduction to Optimum Design

Second Edition

Jasbir S. Arora
The University of Iowa




ELSEVIER
ACADEMIC
PRESS

Amsterdam Boston Heidelberg London New York Oxford
Paris San Diego San Francisco Singapore Sydney Tokyo

Elsevier Academic Press

525 B Street, Suite 1900, San Diego, California 92101-4495, USA
84 Theobald's Road, London WC1X 8RR, UK

This book is printed on acid-free paper. 

Copyright © 2004, Elsevier Inc. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone: (+44) 1865 843830, fax: (+44) 1865 853333, e-mail: permissions@elsevier.com.uk. You may also complete your request on-line via the Elsevier homepage (<http://elsevier.com>), by selecting "Customer Support" and then "Obtaining Permissions."

Library of Congress Cataloging-in-Publication Data

Arora, Jasbir S.

Introduction to optimum design / Jasbir S. Arora.—2nd ed.

p. cm.

Includes bibliographical references and index.

ISBN 0-12-064155-0 (acid-free paper)

1. Engineering design—Mathematical models. I. Title.

TA174.A76 2004

620'.0042'015118—dc22

2004046995

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN: 0-12-064155-0

For all information on all Academic Press publications
visit our Web site at books.elsevier.com

Printed in the United States of America

04 05 06 07 08 09 9 8 7 6 5 4 3 2 1

Jasbir S. Arora

F. Wendell Miller Distinguished Professor of Engineering
Department of Civil and Environmental Engineering
Department of Mechanical and Industrial Engineering
Center for Computer Aided Design
College of Engineering
The University of Iowa
Iowa City, Iowa 52242-1527

To

Ruhee

Rita

Balwant Kaur

Wazir Singh

Preface

I have based the material of the Second Edition on the comments that I had received from the students over the years and on input from colleagues around the world. The text has been rewritten, reorganized, and expanded for the second edition. *Particular attention has been paid to the pedagogical aspect of the material.* Each chapter starts with a list of learning objectives that the students can keep in mind while studying the material of the chapter. The basic philosophy of the text remains the same as before: *to describe an organized approach to engineering design optimization in a rigorous and yet simplified manner, illustrate various concepts and procedures with simple examples, and demonstrate their applicability to engineering design problems.* Formulation of a design problem as an optimization problem is emphasized and illustrated throughout the text. Some computational algorithms are presented in a step-by-step format to give the students a *flavor of the calculations needed for solving optimum design problems.* The *new material* covered in the second edition includes: use of Excel and MATLAB as learning and teaching aids, discrete variable optimization, genetic algorithms, multiobjective optimization, and global optimization.

The text can be used in several ways to design different types of courses for undergraduate and graduate studies. For undergraduate students, the key question is, “*What should be taught on the subject of optimization?*” I feel that the material thoroughly covered should be: optimum design problem formulation, basic concepts that characterize an optimum design, basic concepts of numerical methods for optimization, and simple but illustrative examples of optimum design. In addition, some exposure to the use of optimization software would be quite beneficial. With this background, the students would be able to formulate and use software properly to optimize problems once they go into industry. The basic knowledge gained with this material can serve as a life-long learning tool on the subject of optimum design. *Such a course for junior and senior students* in most branches of engineering can include the following material, augmented with 2- to 3-week-long team projects (*project type exercises and sections with advanced material are marked with an “*” in the text*):

- Appendix A. Economic Analysis
- Chapter 1. Introduction to Design
- Chapter 2. Optimum Design Problem Formulation
- Chapter 3. Graphical Optimization Method
- Chapter 4. Optimum Design Concepts
- Chapter 6. Linear Programming Methods for Optimum Design
- Chapter 8. Numerical Methods for Unconstrained Optimum Design
- Chapter 10. Numerical Methods for Constrained Optimum Design

Another intermediate level course for seniors and first year graduate students can be designed to augment the above material with Chapter 12 on MATLAB along with more advanced design projects and introduction to discrete variable optimization using the material contained in Chapters 15 and 16. The pace of material coverage can be a little faster than the course designed for undergraduates only. A *two-course sequence* for graduate students may be designed using the material from Chapters 1 to 10 and 12 in the first course and the material from Chapters 11 and 13 to 18 for the second course.

I have been fortunate to have received advice, encouragement, and help from numerous people around the globe to undertake and complete this project. Without that, a project of this magnitude would not have been possible. I would like sincerely to thank all of them for their input, in particular, Professor Tae Hee Lee of Hanyang University, and my graduate students Tim Marler and Qian Wang for their *special contributions* to the text material. **Professor Tae Hee Lee** provided me with a first draft of the material for Chapter 12 on Introduction to Optimization with MATLAB. He developed all the examples and the corresponding m-files. **Tim Marler** provided me with first draft of the material for Chapter 17 on Multiobjective Optimum Design Concepts and Methods, and **Qian Wang** provided me with material related to the use of Excel. Without their contributions this material would not be in the good shape it is now. In addition, Tim Marler, Qian Wang, and Ashok Govil proofread several chapters and provided me with suggestions for improving the presentation of the material.

Along with all the individuals mentioned in the first edition, I would like to sincerely thank the following colleagues and friends who provided me with *specific suggestions* on the material for the second edition of the text: Rick Balling, Ashok Belegundu, Scott Burns, Alex Diaz, Dan Frangopol, Ramana Grandhi, Don Grierson, Rafi Haftka, Gene Hou, Tae Hee Lee, T.C. Lin, Kuni Matsui, Duc Nguyen, Makoto Ohsaki, G.J. Park, Subby Rajan, David Thompson, Mats Tinnsten, and Ren-Jye Yang. In addition, the useful exchange of ideas on the subject of optimum design over the years with many colleagues are acknowledged: Santiago Hernández, Hans Eschenauer, Ed Haug, Niels Olhoff, H. Furuta, U. Kirsch, J. Sobieski, Panos Papalambros, Colby Swan, V.K. Goel, F.Y. Cheng, S. Pezeshk, D.H. Choi, Dan Tortorelli, H. Yamakawa, C.M. Chan, Lucien Schmit, V. Kumar, Kwan Rim, Hasan Kamil, Mike Poldneff, Bob Benedict, John Taylor, Marek Rysz, Farrokh Mistree, M.H. Abolbashari, Achille Messac, J. Herskovits, M. Kamat, V. Venkayya, N. Khot, Gary Vanderplaats, B.M. Kwak, George Rozvany, N. Kikuchi, Prabhat Hajela, Z. Gürdal, Nielen Stander, Omar Ghattas, Peter Eriksson, Olof Friberg, Jan Snyman, U. Kirsch, P. Pedersen, K. Truman, C. Mota Soares, Igbal Rai, Rajbir Samra, Jagir Sooch, and many more.

I appreciate my colleagues at The University of Iowa who used the first edition of the book to teach an undergraduate course on optimum design: Karim Abdel-Malek, Asghar Bhatti, Kyung Choi, Ray Han, Harry Kane, George Lance, and Emad Tanbour. Their discussions and suggestions have greatly helped in improving the presentation of the material of first 11 chapters of the second edition.

I would like to acknowledge all my former graduate students whose thesis work on various topics of optimization contributed to the broadening of my horizon on the subject. The recent work of Mike Huang, C.C. Hsieh, Fatos Kocer, and Ossama Elwakeil has formed the basis for the material of Chapters 15, 16, and 18.

I would also like to thank Carla Kinney, Christine Kloiber, Joel Stein, Shoshanna Grossman and Brandy Palacios of Elsevier Science, and Dan Fitzgerald of Graphic World Publishing Services for their support and superb handling of the manuscript for the book.

I am grateful to the Department of Civil and Environmental Engineering, College of Engineering, and The University of Iowa for providing me with time, resources, and support for this very satisfying endeavor.

Finally, I would like to thank all my family and friends for their love and support.

Jasbir Singh Arora
Iowa City

Table of Contents

<i>Preface</i>		<i>ix</i>
Chapter 1	Introduction to Design	1
	1.1 The Design Process	2
	1.2 Engineering Design versus Engineering Analysis	4
	1.3 Conventional versus Optimum Design Process	4
	1.4 Optimum Design versus Optimal Control	6
	1.5 Basic Terminology and Notation	7
	1.5.1 Sets and Points	7
	1.5.2 Notation for Constraints	9
	1.5.3 Superscripts/Subscripts and Summation Notation	9
	1.5.4 Norm/Length of a Vector	11
	1.5.5 Functions	11
	1.5.6 U.S.-British versus SI Units	12
Chapter 2	Optimum Design Problem Formulation	15
	2.1 The Problem Formulation Process	16
	2.1.1 Step 1: Project/Problem Statement	16
	2.1.2 Step 2: Data and Information Collection	16
	2.1.3 Step 3: Identification/Definition of Design Variables	16
	2.1.4 Step 4: Identification of a Criterion to Be Optimized	17
	2.1.5 Step 5: Identification of Constraints	17
	2.2 Design of a Can	18
	2.3 Insulated Spherical Tank Design	20
	2.4 Saw Mill Operation	22
	2.5 Design of a Two-Bar Bracket	24
	2.6 Design of a Cabinet	30
	2.6.1 Formulation 1 for Cabinet Design	30
	2.6.2 Formulation 2 for Cabinet Design	31
	2.6.3 Formulation 3 for Cabinet Design	31

2.7	Minimum Weight Tubular Column Design	32
2.7.1	Formulation 1 for Column Design	33
2.7.2	Formulation 2 for Column Design	34
2.8	Minimum Cost Cylindrical Tank Design	35
2.9	Design of Coil Springs	36
2.10	Minimum Weight Design of a Symmetric Three-Bar Truss	38
2.11	A General Mathematical Model for Optimum Design	41
2.11.1	Standard Design Optimization Model	42
2.11.2	Maximization Problem Treatment	43
2.11.3	Treatment of “Greater Than Type” Constraints	43
2.11.4	Discrete and Integer Design Variables	44
2.11.5	Feasible Set	45
2.11.6	Active/Inactive/Violated Constraints	45
	Exercises for Chapter 2	46
Chapter 3	Graphical Optimization	55
3.1	Graphical Solution Process	55
3.1.1	Profit Maximization Problem	55
3.1.2	Step-by-Step Graphical Solution Procedure	56
3.2	Use of Mathematica for Graphical Optimization	60
3.2.1	Plotting Functions	61
3.2.2	Identification and Hatching of Infeasible Region for an Inequality	62
3.2.3	Identification of Feasible Region	62
3.2.4	Plotting of Objective Function Contours	63
3.2.5	Identification of Optimum Solution	63
3.3	Use of MATLAB for Graphical Optimization	64
3.3.1	Plotting of Function Contours	64
3.3.2	Editing of Graph	64
3.4	Design Problem with Multiple Solutions	66
3.5	Problem with Unbounded Solution	66
3.6	Infeasible Problem	67
3.7	Graphical Solution for Minimum Weight Tubular Column	69
3.8	Graphical Solution for a Beam Design Problem	69
	Exercises for Chapter 3	72
Chapter 4	Optimum Design Concepts	83
4.1	Definitions of Global and Local Minima	84
4.1.1	Minimum	84
4.1.2	Existence of Minimum	89
4.2	Review of Some Basic Calculus Concepts	89
4.2.1	Gradient Vector	90
4.2.2	Hessian Matrix	92
4.2.3	Taylor’s Expansion	93
4.2.4	Quadratic Forms and Definite Matrices	96
4.2.5	Concept of Necessary and Sufficient Conditions	102

4.3	Unconstrained Optimum Design Problems	103
4.3.1	Concepts Related to Optimality Conditions	103
4.3.2	Optimality Conditions for Functions of Single Variable	104
4.3.3	Optimality Conditions for Functions of Several Variables	109
4.3.4	Roots of Nonlinear Equations Using Excel	116
4.4	Constrained Optimum Design Problems	119
4.4.1	Role of Constraints	119
4.4.2	Necessary Conditions: Equality Constraints	121
4.4.3	Necessary Conditions: Inequality Constraints—Karush-Kuhn-Tucker (KKT) Conditions	128
4.4.4	Solution of KKT Conditions Using Excel	140
4.4.5	Solution of KKT Conditions Using MATLAB	141
4.5	Postoptimality Analysis: Physical Meaning of Lagrange Multipliers	143
4.5.1	Effect of Changing Constraint Limits	143
4.5.2	Effect of Cost Function Scaling on Lagrange Multipliers	146
4.5.3	Effect of Scaling a Constraint on Its Lagrange Multiplier	147
4.5.4	Generalization of Constraint Variation Sensitivity Result	148
4.6	Global Optimality	149
4.6.1	Convex Sets	149
4.6.2	Convex Functions	151
4.6.3	Convex Programming Problem	153
4.6.4	Transformation of a Constraint	156
4.6.5	Sufficient Conditions for Convex Programming Problems	157
4.7	Engineering Design Examples	158
4.7.1	Design of a Wall Bracket	158
4.7.2	Design of a Rectangular Beam	162
	Exercises for Chapter 4	166
Chapter 5	More on Optimum Design Concepts	175
5.1	Alternate Form of KKT Necessary Conditions	175
5.2	Irregular Points	178
5.3	Second-Order Conditions for Constrained Optimization	179
5.4	Sufficiency Check for Rectangular Beam Design Problem	184
	Exercises for Chapter 5	185
Chapter 6	Linear Programming Methods for Optimum Design	191
6.1	Definition of a Standard Linear Programming Problem	192
6.1.1	Linear Constraints	192
6.1.2	Unrestricted Variables	193
6.1.3	Standard LP Definition	193

6.2	Basic Concepts Related to Linear Programming Problems	195
6.2.1	Basic Concepts	195
6.2.2	LP Terminology	198
6.2.3	Optimum Solution for LP Problems	201
6.3	Basic Ideas and Steps of the Simplex Method	201
6.3.1	The Simplex	202
6.3.2	Canonical Form/General Solution of $\mathbf{Ax} = \mathbf{b}$	202
6.3.3	Tableau	203
6.3.4	The Pivot Step	205
6.3.5	Basic Steps of the Simplex Method	206
6.3.6	Simplex Algorithm	211
6.4	Two-Phase Simplex Method—Artificial Variables	218
6.4.1	Artificial Variables	219
6.4.2	Artificial Cost Function	219
6.4.3	Definition of Phase I Problem	220
6.4.4	Phase I Algorithm	220
6.4.5	Phase II Algorithm	221
6.4.6	Degenerate Basic Feasible Solution	226
6.5	Postoptimality Analysis	228
6.5.1	Changes in Resource Limits	229
6.5.2	Ranging Right Side Parameters	235
6.5.3	Ranging Cost Coefficients	239
6.5.4	Changes in the Coefficient Matrix	241
6.6	Solution of LP Problems Using Excel Solver	243
	Exercises for Chapter 6	246

Chapter 7	More on Linear Programming Methods for Optimum Design	259
7.1	Derivation of the Simplex Method	259
7.1.1	Selection of a Basic Variable That Should Become Nonbasic	259
7.1.2	Selection of a Nonbasic Variable That Should Become Basic	260
7.2	Alternate Simplex Method	262
7.3	Duality in Linear Programming	263
7.3.1	Standard Primal LP	263
7.3.2	Dual LP Problem	264
7.3.3	Treatment of Equality Constraints	265
7.3.4	Alternate Treatment of Equality Constraints	266
7.3.5	Determination of Primal Solution from Dual Solution	267
7.3.6	Use of Dual Tableau to Recover Primal Solution	271
7.3.7	Dual Variables as Lagrange Multipliers	273
	Exercises for Chapter 7	275

Chapter 8	Numerical Methods for Unconstrained Optimum Design	277
8.1	General Concepts Related to Numerical Algorithms	278
8.1.1	A General Algorithm	279
8.1.2	Descent Direction and Descent Step	280

8.1.3	Convergence of Algorithms	282
8.1.4	Rate of Convergence	282
8.2	Basic Ideas and Algorithms for Step Size Determination	282
8.2.1	Definition of One-Dimensional Minimization Subproblem	282
8.2.2	Analytical Method to Compute Step Size	283
8.2.3	Concepts Related to Numerical Methods to Compute Step Size	285
8.2.4	Equal Interval Search	286
8.2.5	Alternate Equal Interval Search	288
8.2.6	Golden Section Search	289
8.3	Search Direction Determination: Steepest Descent Method	293
8.4	Search Direction Determination: Conjugate Gradient Method	296
	Exercises for Chapter 8	300

Chapter 9	More on Numerical Methods for Unconstrained Optimum Design	305
9.1	More on Step Size Determination	305
9.1.1	Polynomial Interpolation	306
9.1.2	Inaccurate Line Search	309
9.2	More on Steepest Descent Method	310
9.2.1	Properties of the Gradient Vector	310
9.2.2	Orthogonality of Steepest Descent Directions	314
9.3	Scaling of Design Variables	315
9.4	Search Direction Determination: Newton's Method	318
9.4.1	Classical Newton's Method	318
9.4.2	Modified Newton's Method	319
9.4.3	Marquardt Modification	323
9.5	Search Direction Determination: Quasi-Newton Methods	324
9.5.1	Inverse Hessian Updating: DFP Method	324
9.5.2	Direct Hessian Updating: BFGS Method	327
9.6	Engineering Applications of Unconstrained Methods	329
9.6.1	Minimization of Total Potential Energy	329
9.6.2	Solution of Nonlinear Equations	331
9.7	Solution of Constrained Problems Using Unconstrained Optimization Methods	332
9.7.1	Sequential Unconstrained Minimization Techniques	333
9.7.2	Multiplier (Augmented Lagrangian) Methods	334
	Exercises for Chapter 9	335

Chapter 10	Numerical Methods for Constrained Optimum Design	339
10.1	Basic Concepts and Ideas	340
10.1.1	Basic Concepts Related to Algorithms for Constrained Problems	340
10.1.2	Constraint Status at a Design Point	342
10.1.3	Constraint Normalization	343

10.1.4	Descent Function	345
10.1.5	Convergence of an Algorithm	345
10.2	Linearization of Constrained Problem	346
10.3	Sequential Linear Programming Algorithm	352
10.3.1	The Basic Idea—Move Limits	352
10.3.2	An SLP Algorithm	353
10.3.3	SLP Algorithm: Some Observations	357
10.4	Quadratic Programming Subproblem	358
10.4.1	Definition of QP Subproblem	358
10.4.2	Solution of QP Subproblem	361
10.5	Constrained Steepest Descent Method	363
10.5.1	Descent Function	364
10.5.2	Step Size Determination	366
10.5.3	CSD Algorithm	368
10.5.4	CSD Algorithm: Some Observations	368
10.6	Engineering Design Optimization Using Excel Solver	369
	Exercises for Chapter 10	373

Chapter 11

	More on Numerical Methods for Constrained Optimum Design	379
11.1	Potential Constraint Strategy	379
11.2	Quadratic Programming Problem	383
11.2.1	Definition of QP Problem	383
11.2.2	KKT Necessary Conditions for the QP Problem	384
11.2.3	Transformation of KKT Conditions	384
11.2.4	Simplex Method for Solving QP Problem	385
11.3	Approximate Step Size Determination	388
11.3.1	The Basic Idea	388
11.3.2	Descent Condition	389
11.3.3	CSD Algorithm with Approximate Step Size	393
11.4	Constrained Quasi-Newton Methods	400
11.4.1	Derivation of Quadratic Programming Subproblem	400
11.4.2	Quasi-Newton Hessian Approximation	403
11.4.3	Modified Constrained Steepest Descent Algorithm	404
11.4.4	Observations on the Constrained Quasi-Newton Methods	406
11.4.5	Descent Functions	406
11.5	Other Numerical Optimization Methods	407
11.5.1	Method of Feasible Directions	407
11.5.2	Gradient Projection Method	409
11.5.3	Generalized Reduced Gradient Method	410
	Exercises for Chapter 11	411

Chapter 12

	Introduction to Optimum Design with MATLAB	413
12.1	Introduction to Optimization Toolbox	413
12.1.1	Variables and Expressions	413

12.1.2	Scalar, Array, and Matrix Operations	414
12.1.3	Optimization Toolbox	414
12.2	Unconstrained Optimum Design Problems	415
12.3	Constrained Optimum Design Problems	418
12.4	Optimum Design Examples with MATLAB	420
12.4.1	Location of Maximum Shear Stress for Two Spherical Bodies in Contact	420
12.4.2	Column Design for Minimum Mass	421
12.4.3	Flywheel Design for Minimum Mass	425
	Exercises for Chapter 12	429

Chapter 13	Interactive Design Optimization	433
13.1	Role of Interaction in Design Optimization	434
13.1.1	What Is Interactive Design Optimization?	434
13.1.2	Role of Computers in Interactive Design Optimization	434
13.1.3	Why Interactive Design Optimization?	435
13.2	Interactive Design Optimization Algorithms	436
13.2.1	Cost Reduction Algorithm	436
13.2.2	Constraint Correction Algorithm	440
13.2.3	Algorithm for Constraint Correction at Constant Cost	442
13.2.4	Algorithm for Constraint Correction at Specified Increase in Cost	445
13.2.5	Constraint Correction with Minimum Increase in Cost	446
13.2.6	Observations on Interactive Algorithms	447
13.3	Desired Interactive Capabilities	448
13.3.1	Interactive Data Preparation	448
13.3.2	Interactive Capabilities	448
13.3.3	Interactive Decision Making	449
13.3.4	Interactive Graphics	450
13.4	Interactive Design Optimization Software	450
13.4.1	User Interface for IDESIGN	451
13.4.2	Capabilities of IDESIGN	453
13.5	Examples of Interactive Design Optimization	454
13.5.1	Formulation of Spring Design Problem	454
13.5.2	Optimum Solution for the Spring Design Problem	455
13.5.3	Interactive Solution for Spring Design Problem	455
13.5.4	Use of Interactive Graphics	457
	Exercises for Chapter 13	462

Chapter 14	Design Optimization Applications with Implicit Functions	465
14.1	Formulation of Practical Design Optimization Problems	466
14.1.1	General Guidelines	466
14.1.2	Example of a Practical Design Optimization Problem	467

14.2	Gradient Evaluation for Implicit Functions	473
14.3	Issues in Practical Design Optimization	478
14.3.1	Selection of an Algorithm	478
14.3.2	Attributes of a Good Optimization Algorithm	478
14.4	Use of General-Purpose Software	479
14.4.1	Software Selection	480
14.4.2	Integration of an Application into General-Purpose Software	480
14.5	Optimum Design of Two-Member Frame with Out-of-Plane Loads	481
14.6	Optimum Design of a Three-Bar Structure for Multiple Performance Requirements	483
14.6.1	Symmetric Three-Bar Structure	483
14.6.2	Asymmetric Three-Bar Structure	484
14.6.3	Comparison of Solutions	490
14.7	Discrete Variable Optimum Design	491
14.7.1	Continuous Variable Optimization	492
14.7.2	Discrete Variable Optimization	492
14.8	Optimal Control of Systems by Nonlinear Programming	493
14.8.1	A Prototype Optimal Control Problem	493
14.8.2	Minimization of Error in State Variable	497
14.8.3	Minimum Control Effort Problem	503
14.8.4	Minimum Time Control Problem	505
14.8.5	Comparison of Three Formulations for Optimal Control of System Motion	508
	Exercises for Chapter 14	508

Chapter 15	Discrete Variable Optimum Design Concepts and Methods	513
15.1	Basic Concepts and Definitions	514
15.1.1	Definition of Mixed Variable Optimum Design Problem: MV-OPT	514
15.1.2	Classification of Mixed Variable Optimum Design Problems	514
15.1.3	Overview of Solution Concepts	515
15.2	Branch and Bound Methods (BBM)	516
15.2.1	Basic BBM	517
15.2.2	BBM with Local Minimization	519
15.2.3	BBM for General MV-OPT	520
15.3	Integer Programming	521
15.4	Sequential Linearization Methods	522
15.5	Simulated Annealing	522
15.6	Dynamic Rounding-off Method	524
15.7	Neighborhood Search Method	525
15.8	Methods for Linked Discrete Variables	525
15.9	Selection of a Method	526
	Exercises for Chapter 15	527

Chapter 16	Genetic Algorithms for Optimum Design	531
16.1	Basic Concepts and Definitions	532
16.2	Fundamentals of Genetic Algorithms	534

16.3	Genetic Algorithm for Sequencing-Type Problems	538
16.4	Applications	539
	Exercises for Chapter 16	540
Chapter 17	Multiobjective Optimum Design Concepts and Methods	543
17.1	Problem Definition	543
17.2	Terminology and Basic Concepts	546
17.2.1	Criterion Space and Design Space	546
17.2.2	Solution Concepts	548
17.2.3	Preferences and Utility Functions	551
17.2.4	Vector Methods and Scalarization Methods	551
17.2.5	Generation of Pareto Optimal Set	551
17.2.6	Normalization of Objective Functions	552
17.2.7	Optimization Engine	552
17.3	Multiobjective Genetic Algorithms	552
17.4	Weighted Sum Method	555
17.5	Weighted Min-Max Method	556
17.6	Weighted Global Criterion Method	556
17.7	Lexicographic Method	558
17.8	Bounded Objective Function Method	558
17.9	Goal Programming	559
17.10	Selection of Methods	559
	Exercises for Chapter 17	560
Chapter 18	Global Optimization Concepts and Methods for Optimum Design	565
18.1	Basic Concepts of Solution Methods	565
18.1.1	Basic Concepts	565
18.1.2	Overview of Methods	567
18.2	Overview of Deterministic Methods	567
18.2.1	Covering Methods	568
18.2.2	Zooming Method	568
18.2.3	Methods of Generalized Descent	569
18.2.4	Tunneling Method	571
18.3	Overview of Stochastic Methods	572
18.3.1	Pure Random Search	573
18.3.2	Multistart Method	573
18.3.3	Clustering Methods	573
18.3.4	Controlled Random Search	575
18.3.5	Acceptance-Rejection Methods	578
18.3.6	Stochastic Integration	579
18.4	Two Local-Global Stochastic Methods	579
18.4.1	A Conceptual Local-Global Algorithm	579
18.4.2	Domain Elimination Method	580
18.4.3	Stochastic Zooming Method	582
18.4.4	Operations Analysis of the Methods	583
18.5	Numerical Performance of Methods	585
18.5.1	Summary of Features of Methods	585
18.5.2	Performance of Some Methods Using Unconstrained Problems	586

	18.5.3 Performance of Stochastic Zooming and Domain Elimination Methods	586
	18.5.4 Global Optimization of Structural Design Problems	587
	Exercises for Chapter 18	588
<i>Appendix A</i>	<i>Economic Analysis</i>	593
A.1	Time Value of Money	593
	A.1.1 Cash Flow Diagrams	594
	A.1.2 Basic Economic Formulas	594
A.2	Economic Bases for Comparison	598
	A.2.1 Annual Base Comparisons	599
	A.2.2 Present Worth Comparisons	601
	Exercises for Appendix A	604
<i>Appendix B</i>	<i>Vector and Matrix Algebra</i>	611
B.1	Definition of Matrices	611
B.2	Type of Matrices and Their Operations	613
	B.2.1 Null Matrix	613
	B.2.2 Vector	613
	B.2.3 Addition of Matrices	613
	B.2.4 Multiplication of Matrices	613
	B.2.5 Transpose of a Matrix	615
	B.2.6 Elementary Row–Column Operations	616
	B.2.7 Equivalence of Matrices	616
	B.2.8 Scalar Product–Dot Product of Vectors	616
	B.2.9 Square Matrices	616
	B.2.10 Partitioning of Matrices	617
B.3	Solution of n Linear Equations in n Unknowns	618
	B.3.1 Linear Systems	618
	B.3.2 Determinants	619
	B.3.3 Gaussian Elimination Procedure	621
	B.3.4 Inverse of a Matrix: Gauss-Jordan Elimination	625
B.4	Solution of m Linear Equations in n Unknowns	628
	B.4.1 Rank of a Matrix	628
	B.4.2 General Solution of $m \times n$ Linear Equations	629
B.5	Concepts Related to a Set of Vectors	635
	B.5.1 Linear Independence of a Set of Vectors	635
	B.5.2 Vector Spaces	639
B.6	Eigenvalues and Eigenvectors	642
B.7	Norm and Condition Number of a Matrix	643
	B.7.1 Norm of Vectors and Matrices	643
	B.7.2 Condition Number of a Matrix	644
	Exercises for Appendix B	645
<i>Appendix C</i>	<i>A Numerical Method for Solution of Nonlinear Equations</i>	647
C.1	Single Nonlinear Equation	647
C.2	Multiple Nonlinear Equations	650
	Exercises for Appendix C	655

<i>Appendix D</i>	<i>Sample Computer Programs</i>	657
	D.1 Equal Interval Search	657
	D.2 Golden Section Search	660
	D.3 Steepest Descent Method	660
	D.4 Modified Newton's Method	669
<i>References</i>		675
<i>Bibliography</i>		683
<i>Answers to Selected Problems</i>		687
<i>Index</i>		695

1 Introduction to Design

Upon completion of this chapter, you will be able to:

- Describe the overall process of designing systems
- Distinguish between engineering design and engineering analysis activity
- Distinguish between the conventional design process and optimum design process
- Distinguish between the optimum design and optimal control problems
- Understand the notations used for operations with vectors, matrices, and functions

Engineering consists of a number of well established activities, including analysis, design, fabrication, sales, research, and the development of systems. The subject of this text—the design of systems—is a major field in the engineering profession. The process of designing and fabricating systems has been developed over centuries. The existence of many complex systems, such as buildings, bridges, highways, automobiles, airplanes, space vehicles, and others, is an excellent testimonial for this process. However, the evolution of these systems has been slow. The entire process has been both time-consuming and costly, requiring substantial human and material resources. Therefore, the procedure has been to design, fabricate, and use the system regardless of whether it was the *best one*. Improved systems were designed only after a substantial investment had been recovered. These new systems performed the same or even more tasks, cost less, and were more efficient.

The preceding discussion indicates that several systems can usually accomplish the same task, and that some are better than others. For example, the purpose of a bridge is to provide continuity in traffic from one side to the other. Several types of bridges can serve this purpose. However, to analyze and design all possibilities can be a time-consuming and costly affair. Usually one type has been selected based on some preliminary analyses and has been designed in detail.

The design of complex systems requires data processing and a large number of calculations. In the recent past, a revolution in computer technology and numerical computations has taken place. Today's computers can perform complex calculations and process large amounts of data rapidly. The engineering design and optimization processes benefit greatly from this revolution because they require a large number of calculations. Better systems can now be designed by analyzing and optimizing various options in a short time. This is highly

desirable because better designed systems cost less, have more capability, and are easy to maintain and operate.

The design of systems can be *formulated as problems of optimization* in which a measure of performance is to be optimized while satisfying all constraints. Many numerical methods of optimization have been developed and used to design better systems. This text describes the basic concepts of optimization methods and their applications to the design of engineering systems. Design process is emphasized rather than optimization theory. Various theorems are stated as results without rigorous proofs. However, their implications from an engineering point of view are studied and discussed in detail. Optimization theory, numerical methods, and modern computer hardware and software can be used as tools to design better engineering systems. The text emphasizes this theme throughout.

Any problem in which certain parameters need to be determined to satisfy constraints can be formulated as an optimization problem. Once this has been done, the concepts and the methods described in this text can be used to solve the problem. Therefore, the optimization techniques are quite general, having a wide range of applicability in diverse fields. The range of applications is limited only by the imagination or ingenuity of the designers. It is impossible to discuss every application of optimization concepts and techniques in this introductory text. However, using simple applications, we shall discuss concepts, fundamental principles, and basic techniques that can be used in numerous applications. The student should understand them without getting bogged down with the notation, terminology, and details of the particular area of application.

1.1 The Design Process

How do I begin to design a system?

The design of many engineering systems can be a fairly complex process. Many assumptions must be made to develop models that can be subjected to analysis by the available methods and the models must be verified by experiments. Many possibilities and factors must be considered during the problem formulation phase. *Economic considerations* play an important role in designing cost-effective systems. Introductory methods of economic analysis described in Appendix A are useful in this regard. To complete the design of an engineering system, designers from different fields of engineering must usually cooperate. For example, the design of a high-rise building involves designers from architectural, structural, mechanical, electrical, and environmental engineering as well as construction management experts. Design of a passenger car requires cooperation among structural, mechanical, automotive, electrical, human factors, chemical, and hydraulics design engineers. Thus, in an *interdisciplinary environment* considerable interaction is needed among various design teams to complete the project. For most applications the entire design project must be broken down into several subproblems which are then treated independently. Each of the subproblems can be posed as a problem of optimum design.

The design of a system begins by analyzing various options. Subsystems and their components are identified, designed, and tested. This process results in a set of drawings, calculations, and reports by which the system can be fabricated. We shall use a systems engineering model to describe the *design process*. Although a complete discussion of this subject is beyond the scope of the text, some basic concepts will be discussed using a simple block diagram.

Design is an *iterative process*. The designer's experience, intuition, and ingenuity are required in the design of systems in most fields of engineering (aerospace, automotive, civil, chemical, industrial, electrical, mechanical, hydraulic, and transportation). *Iterative* implies analyzing several *trial designs* one after another until an acceptable design is obtained. The concept of trial designs is important to understand. In the design process, the designer

estimates a trial design of the system based on experience, intuition, or some mathematical analysis. The trial design is analyzed to determine if it is acceptable. If it is, the design process is terminated. In the optimization process, the trial design is analyzed to determine if it is the best. Depending on the specifications, “best” can have different connotations for different systems. In general, it implies cost-effective, efficient, reliable and durable systems. The process can require considerable interaction among teams of specialists from different disciplines. The basic concepts are described in the text to aid the engineer in designing systems at the minimum cost and in the shortest amount of time.

The design process should be a well organized activity. To discuss it, we consider a *system evolution model* shown in Fig. 1-1. The process begins with the identification of a need which may be conceived by engineers or nonengineers.

The *first step* in the evolutionary process is to define precisely specifications for the system. Considerable interaction between the engineer and the sponsor of the project is usually necessary to quantify the *system specifications*. Once these are identified, the task of designing the system can begin.

The *second step* in the process is to develop a *preliminary design* of the system. Various concepts for the system are studied. Since this must be done in a relatively short time, *simplified models* are used. Various subsystems are identified and their preliminary designs estimated. Decisions made at this stage generally affect the final appearance and performance of the system. At the end of the preliminary design phase, a few promising concepts that need further analysis are identified.

The *third step* in the process is to carry out a *detailed design* for all subsystems using an iterative process. To evaluate various possibilities, this must be done for all previously identified promising concepts. The design parameters for the subsystems must be identified. The system performance requirements must be identified and satisfied. The subsystems must be designed to maximize system worth or to minimize a measure of the cost. Systematic optimization methods described in this text can aid the designer in accelerating the detailed design process. At the end of the process, a description of the system is available in the form of reports and drawings.

The *fourth and fifth blocks* of Fig. 1-1 may or may not be necessary for all systems. They involve fabrication of a prototype system and testing. These steps are necessary when the system has to be mass produced or when human lives are involved. Although these blocks may appear to be the final steps in the design process, they are not because the system may not perform according to specifications during the testing phase. Therefore, specifications may have to be modified or other concepts may have to be studied. In fact, this re-examination may be necessary at any step in the design process. It is for this reason that *feedback loops* are placed at every stage of the system evolution process, as shown in Fig. 1-1. The iterative process has to be continued until an acceptable system has evolved. Depending on the complexity of the system, the process may take a few days or several months.

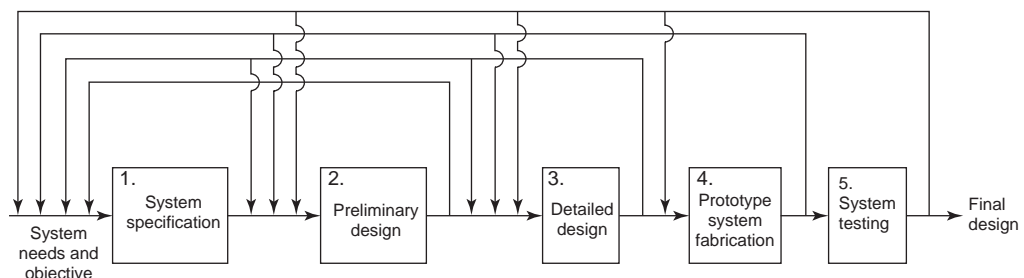


FIGURE 1-1 A system evolution model.

The previously described model is a simplified block diagram for system evolution. In actual practice, each block may have to be broken down into several sub-blocks to carry out the studies properly and arrive at rational decisions. *The important point is that optimization concepts and methods can help at every stage in the process.* The use of such methods along with appropriate software can be extremely useful in studying various design possibilities rapidly. These techniques can be useful during preliminary and detailed design phases as well as for fabrication and testing. Therefore, in this text, we discuss optimization methods and their use in the design process.

At some stages in the design process, it may appear that the process can be completely automated, that the designer can be eliminated from the loop, and that optimization methods and programs can be used as black boxes. This may be true in some cases. However, the design of a system is a creative process that can be quite complex. It may be ill-defined and a solution to the design problem may not exist. Problem functions may not be defined in certain regions of the design space. Thus, for most practical problems, designers play a key role in guiding the process to acceptable regions. Designers must be an integral part of the process and use their intuition and judgment in obtaining the final design. More details of the interactive design optimization process and the role of the designer are discussed in Chapter 13.

1.2 Engineering Design versus Engineering Analysis

Can I design without analysis?

It is important to recognize differences between *engineering analysis* and *design activities*. The analysis problem is concerned with determining the behavior of an existing system or a trial system being designed for a given task. Determination of the behavior of the system implies calculation of its response under specified inputs. Therefore, the sizes of various parts and their configurations are given for the analysis problem, i.e., the design of the system is known. On the other hand, the design process calculates the sizes and shapes of various parts of the system to meet performance requirements. The design of a system is a trial and error procedure. We estimate a design and analyze it to see if it performs according to given specifications. If it does, we have an *acceptable (feasible) design*, although we may still want to change it to improve its performance. If the trial design does not work, we need to change it to come up with an acceptable system. In both cases, we must be able to *analyze designs* to make further decisions. Thus, analysis capability must be available in the design process.

This book is intended for use in all branches of engineering. It is assumed throughout that students understand analysis methods covered in undergraduate engineering statics and physics courses. However, *we will not let the lack of analysis capability hinder the understanding of the systematic process of optimum design.* Equations for analysis of the system will be given wherever needed.

1.3 Conventional versus Optimum Design Process

Why do I want to optimize?

It is a challenge for engineers to design efficient and cost-effective systems without compromising the integrity of the system. The conventional design process depends on the designer's intuition, experience, and skill. This presence of a human element can sometimes lead to erroneous results in the synthesis of complex systems. Figure 1-2(A) presents a self-explanatory flowchart for a conventional design process that involves the use of information gathered from one or more trial designs together with the designer's experience and intuition.

Because you want to beat the competition and improve your bottom line!

Scarcity and the need for efficiency in today's competitive world have forced engineers to evince greater interest in economical and better designs. The computer-aided design optimization (CADO) process can help in this regard. Figure 1-2(B) shows the optimum design process. Both conventional and optimum design processes can be used at different stages of system evolution. The main advantage in the conventional design process is that the designer's experience and intuition can be used in making conceptual changes in the system or to make additional specifications in the procedure. For example, the designer can choose either a suspension bridge or an arched bridge, add or delete certain components of the structure, and so on. When it comes to detailed design, however, the conventional design process has some disadvantages. These include the treatment of complex constraints (such as limits on vibration frequencies) as well as inputs (for example, when the structure is subjected to a variety of loading conditions). In these cases, the designer would find it difficult to decide whether to increase or decrease the size of a particular structural element to satisfy the constraints. Furthermore, the conventional design process can lead to uneconomical designs and can involve a lot of calendar time. *The optimum design process forces the designer to identify explicitly a set of design variables, an objective function to be optimized, and the constraint functions for the system.* This rigorous formulation of the design problem helps the designer gain a better understanding of the problem. Proper mathematical formulation of the design problem is a key to good solutions. This topic is discussed in more detail in Chapter 2.

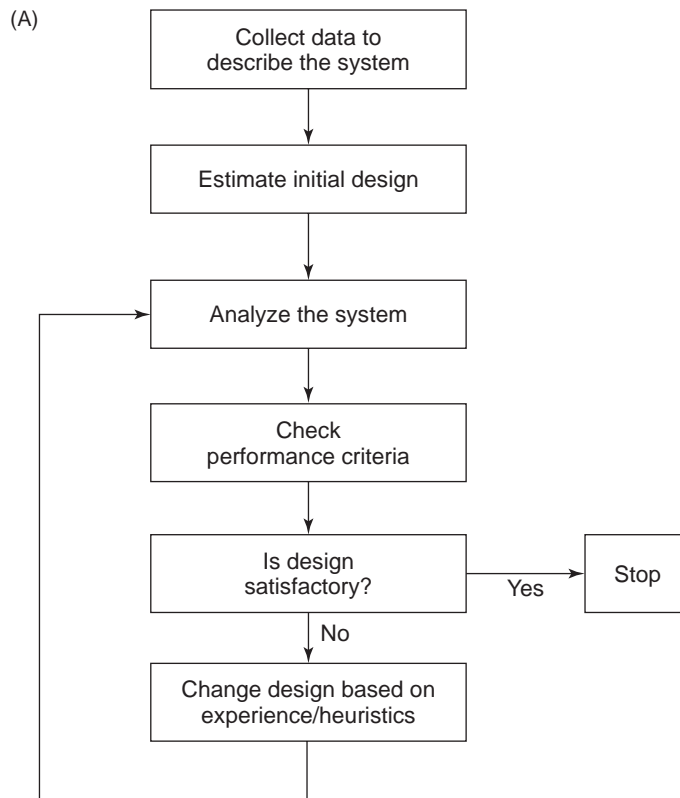


FIGURE 1-2 Comparison of conventional and optimum design processes. (A) Conventional design process.

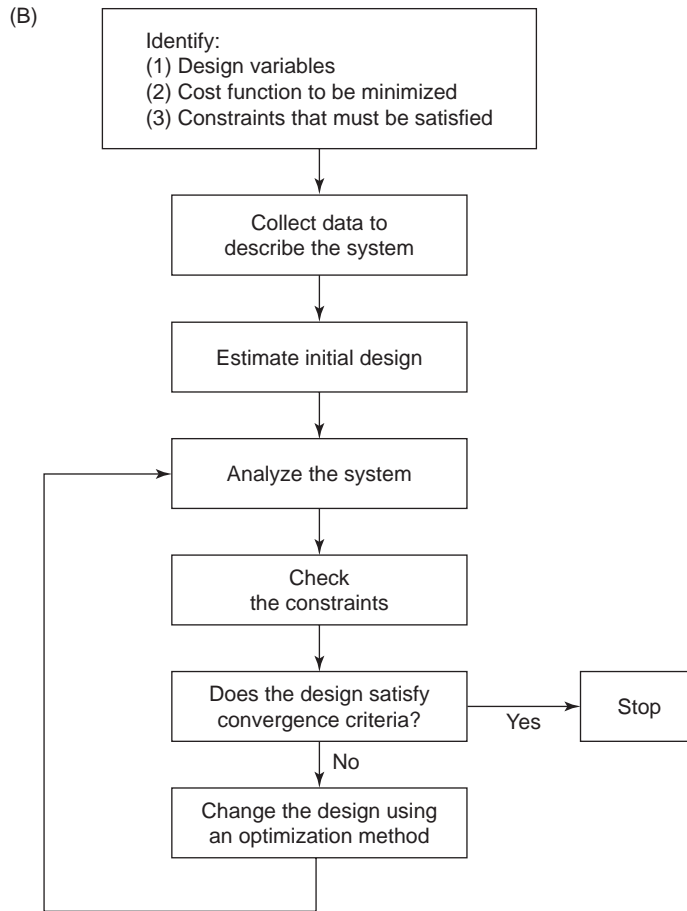


FIGURE 1-2 continued (B) Optimum design process.

The foregoing distinction between the two design approaches indicates that the conventional design process is less formal. An objective function that measures the performance of the system is not identified. Trend information is not calculated to make design decisions for improvement of the system. Most decisions are made based on the designer's experience and intuition. In contrast, the optimization process is more formal, using trend information to make decisions. However, the optimization process can substantially benefit from the designer's experience and intuition in formulating the problem and identifying the critical constraints. Thus, the best approach would be an optimum design process that is aided by the designer's interaction.

1.4 Optimum Design versus Optimal Control

Optimum design and optimal control of systems are two separate activities. There are numerous applications in which methods of optimum design are useful in designing systems. There are many other applications where optimal control concepts are needed. In addition, there are some applications in which both optimum design and optimal control concepts must be used. Sample applications include *robotics* and *aerospace structures*. In this text, optimal

control problems and methods are not described in detail. However, fundamental differences between the two activities are briefly explained in the sequel. It turns out that optimal control problems can be transformed into optimum design problems and treated by the methods described in the text. Thus, methods of optimum design are very powerful and should be clearly understood. A simple optimal control problem is described in Chapter 14 and is solved by the methods of optimum design.

The optimal control problem consists of finding feedback controllers for a system to produce the desired output. The system has active elements that sense fluctuations in the output. System controls are automatically adjusted to correct the situation and optimize a measure of performance. Thus, control problems are usually dynamic in nature. In optimum design, on the other hand, we design the system and its elements to optimize an objective function. The system then remains fixed for its entire life.

As an example, consider the cruise control mechanism in passenger cars. The idea behind this feedback system is to control fuel injection to maintain a constant speed. Thus the system's output is known, i.e., the cruising speed of the car. The job of the control mechanism is to sense fluctuations in the speed depending upon road conditions and to adjust fuel injection accordingly.

1.5 Basic Terminology and Notation

What notation do I need to know?

To understand and be comfortable with the methods of optimum design, familiarity with linear algebra (vector and matrix operations) and basic calculus is essential. Operations of *linear algebra* are described in Appendix B. Students who are not comfortable with that material must review it thoroughly. Calculus of functions of single and multiple variables must also be understood. These concepts are reviewed wherever they are needed. In this section, the *standard terminology* and *notations* used throughout the text are defined. It is important to understand and memorize these, because without them it will be difficult to follow the rest of the text.

1.5.1 Sets and Points

Since realistic systems generally involve several variables, it is necessary to define and utilize some convenient and compact notation. *Set* and *vector notations* serve this purpose quite well and are utilized throughout the text. The terms *vector* and *point* are used interchangeably and *lowercase letters in boldface* are used to denote them. *Upper case letters in boldface* represent *matrices*.

A *point* means an ordered list of numbers. Thus, (x_1, x_2) is a point consisting of the two numbers; (x_1, x_2, \dots, x_n) is a point consisting of the n numbers. Such a point is often called an n -tuple. Each of the numbers is called a component of the (point) vector. Thus, x_1 is the first component, x_2 is the second, and so forth. The n components x_1, x_2, \dots, x_n can be collected into a column vector as

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = [x_1 \ x_2 \ \dots \ x_n]^T \quad (1.1a)$$

where the superscript T denotes the *transpose* of a vector or a matrix, a notation that is used throughout the text (refer to Appendix B for a detailed discussion of vector and matrix algebra). We shall also use the notation

$$\mathbf{x} = (x_1, x_2, \dots, x_n) \quad (1.1b)$$

to represent a point or vector in n -dimensional space. This is called an n -vector.

In three-dimensional space, the vector $\mathbf{x} = [x_1 \ x_2 \ x_3]^T$ represents a point P as shown in Fig. 1-3. Similarly, when there are n components in a vector, as in Eqs. (1.1a) and (1.1b), \mathbf{x} is interpreted as a point in the n -dimensional real space denoted as R^n . The space R^n is simply the collection of all n -vectors (points) of real numbers. For example, the real line is R^1 and the plane is R^2 , and so on.

Often we deal with *sets* of points satisfying certain conditions. For example, we may consider a set S of all points having three components with the last component being zero, which is written as

$$S = \{\mathbf{x} = (x_1, x_2, x_3) \mid x_3 = 0\} \quad (1.2)$$

Information about the set is contained in braces. Equation (1.2) reads as “ S equals the set of all points (x_1, x_2, x_3) with $x_3 = 0$.” The vertical bar divides information about the set S into two parts: to the left of the bar is the dimension of points in the set; to the right are the properties that distinguish those points from others not in the set (for example, properties a point must possess to be in the set S).

Members of a set are sometimes called *elements*. If a point \mathbf{x} is an element of the set S , then we write $\mathbf{x} \in S$. The expression “ $\mathbf{x} \in S$ ” is read, “ \mathbf{x} is an element of (belongs to) S .” Conversely, the expression “ $\mathbf{y} \notin S$ ” is read, “ \mathbf{y} is not an element of (does not belong to) S .”

If all the elements of a set S are also elements of another set T , then S is said to be a “subset of T .” Symbolically, we write $S \subset T$ which is read as, “ S is a subset of T ,” or “ S is contained in T .” Alternatively, we say T is a superset of S , written as $T \supset S$.

As an example of a set S , consider a domain of the x_1 - x_2 plane enclosed by a circle of radius 3 with the center at the point $(4, 4)$, as shown in Fig. 1-4. Mathematically, all points within and on the circle can be expressed as

$$S = \{\mathbf{x} \in R^2 \mid (x_1 - 4)^2 + (x_2 - 4)^2 \leq 9\} \quad (1.3)$$

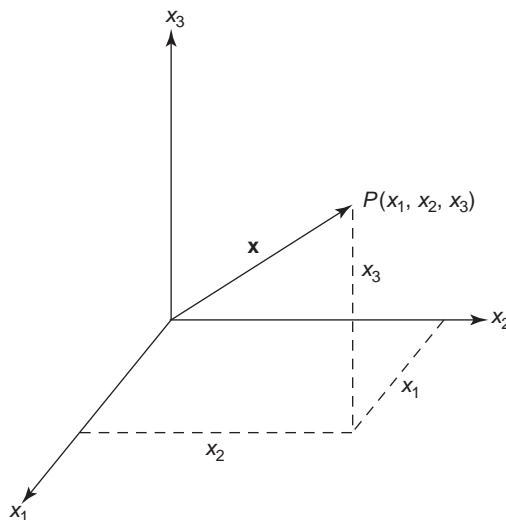


FIGURE 1-3 Vector representation of a point P in three-dimensional space.

Thus, the center of the circle (4, 4) is in the set S because it satisfies the inequality in Eq. (1.3). We write this as $(4, 4) \in S$. The origin of coordinates (0, 0) does not belong to the set since it does not satisfy the inequality in Eq. (1.3). We write this as $(0, 0) \notin S$. It can be verified that the following points belong to the set: (3, 3), (2, 2), (3, 2), (6, 6). In fact, set S has an infinite number of points. Many other points are not in the set. It can be verified that the following points are not in the set: (1, 1), (8, 8), (-1, 2).

1.5.2 Notation for Constraints

Constraints arise naturally in optimum design problems. For example, material of the system must not fail, the demand must be met, resources must not be exceeded, and so on. We shall discuss the constraints in more detail in Chapter 2. Here we discuss the terminology and notations for the constraints.

We have already encountered a constraint in Fig. 1-4 that shows a set S of points within and on the circle of radius 3. The set S is defined by the following constraint:

$$(x_1 - 4)^2 + (x_2 - 4)^2 \leq 9$$

A constraint of this form will be called a *less than or equal to type*. It shall be abbreviated as “ \leq type.” Similarly, there are *greater than or equal to type* constraints, abbreviated as “ \geq type.” Both types are called *inequality constraints*.

1.5.3 Superscripts/Subscripts and Summation Notation

Later we will discuss a set of vectors, components of vectors, and multiplication of matrices and vectors. To write such quantities in a convenient form, consistent and compact notations must be used. We define these notations here. *Superscripts are used to represent different vectors and matrices*. For example, $\mathbf{x}^{(i)}$ represents the i th vector of a set, and $\mathbf{A}^{(k)}$ represents the k th matrix. *Subscripts are used to represent components of vectors and matrices*. For example, x_j is the j th component of \mathbf{x} and a_{ij} is the i - j element of matrix \mathbf{A} . Double subscripts are used to denote elements of a matrix.

To indicate the *range of a subscript* or superscript we use the notation

$$x_i; i = 1 \text{ to } n \tag{1.4}$$

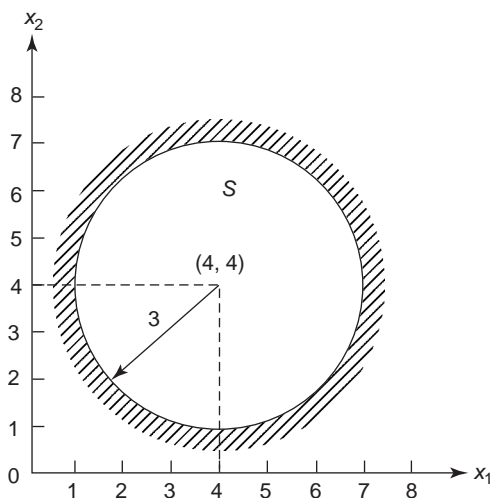


FIGURE 1-4 Geometrical representation for the set $S = \{\mathbf{x} \mid (x_1 - 4)^2 + (x_2 - 4)^2 \leq 9\}$.

This represents the numbers x_1, x_2, \dots, x_n . Note that “ $i = 1$ to n ” represents the range for the index i and is read, “ i goes from 1 to n .” Similarly, a set of k vectors, each having n components, will be represented as

$$\mathbf{x}^{(j)}; j = 1 \text{ to } k \quad (1.5)$$

This represents the k vectors $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(k)}$. It is important to note that subscript i in Eq. (1.4) and superscript j in Eq. (1.5) are *free indices*, i.e., they can be replaced by any other variable. For example, Eq. (1.4) can also be written as $x_j; j = 1$ to n and Eq. (1.5) can be written as $\mathbf{x}^{(i)}; i = 1$ to k . Note that the superscript j in Eq. (1.5) does not represent the power of \mathbf{x} . It is an index that represents the j th vector of a set of vectors.

We shall also use the *summation notation* quite frequently. For example,

$$c = x_1y_1 + x_2y_2 + \dots + x_ny_n \quad (1.6)$$

will be written as

$$c = \sum_{i=1}^n x_iy_i \quad (1.7)$$

Also, multiplication of an n -dimensional vector \mathbf{x} by an $m \times n$ matrix \mathbf{A} to obtain an m -dimensional vector \mathbf{y} , is written as

$$\mathbf{y} = \mathbf{A}\mathbf{x} \quad (1.8)$$

Or, in summation notation, the i th component of \mathbf{y} is

$$y_i = \sum_{j=1}^n a_{ij}x_j = a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n; i = 1 \text{ to } m \quad (1.9)$$

There is another way of writing the matrix multiplication of Eq. (1.8). Let m -dimensional vectors $\mathbf{a}^{(i)}; i = 1$ to n represent columns of the matrix \mathbf{A} . Then, $\mathbf{y} = \mathbf{A}\mathbf{x}$ is also given as

$$\mathbf{y} = \sum_{j=1}^n \mathbf{a}^{(j)}x_j = \mathbf{a}^{(1)}x_1 + \mathbf{a}^{(2)}x_2 + \dots + \mathbf{a}^{(n)}x_n \quad (1.10)$$

The sum on the right side of Eq. (1.10) is said to be a *linear combination* of columns of the matrix \mathbf{A} with $x_j, j = 1$ to n as multipliers of the linear combination. Or, \mathbf{y} is given as a linear combination of columns of \mathbf{A} (refer to Appendix B for further discussion on the linear combination of vectors).

Occasionally, we will have to use the double summation notation. For example, assuming $m = n$ and substituting y_i from Eq. (1.9) into Eq. (1.7), we obtain the double sum as

$$c = \sum_{i=1}^n x_i \left(\sum_{j=1}^n a_{ij}x_j \right) = \sum_{i=1}^n \sum_{j=1}^n a_{ij}x_ix_j \quad (1.11)$$

Note that the indices i and j in Eq. (1.11) can be interchanged. This is possible because c is a *scalar quantity*, so its value is not affected by whether we first sum on i or j . Equation (1.11) can also be written in the matrix form as will be shown later.

1.5.4 Norm/Length of a Vector

If we let \mathbf{x} and \mathbf{y} be two n -dimensional vectors, then their *dot product* is defined as

$$(\mathbf{x} \cdot \mathbf{y}) = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^n x_i y_i \quad (1.12)$$

Thus, the dot product is a sum of the product of corresponding elements of the vectors \mathbf{x} and \mathbf{y} . Two vectors are said to be *orthogonal (normal)* if their dot product is zero, i.e., \mathbf{x} and \mathbf{y} are orthogonal if $\mathbf{x} \cdot \mathbf{y} = 0$. If the vectors are not orthogonal, the angle between them can be calculated from the definition of the dot product:

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos \theta \quad (1.13)$$

where θ is the angle between vectors \mathbf{x} and \mathbf{y} , and $\|\mathbf{x}\|$ represents the *length of the vector* \mathbf{x} . This is also called the *norm of the vector* (for a more general definition of the norm, refer to Appendix B). The length of a vector \mathbf{x} is defined as the square root of the sum of squares of the components, i.e.,

$$\|\mathbf{x}\| = \sqrt{\sum_{i=1}^n x_i^2} = \sqrt{\mathbf{x} \cdot \mathbf{x}} \quad (1.14)$$

The double sum of Eq. (1.11) can be written in the matrix form as follows:

$$c = \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i x_j = \sum_{i=1}^n x_i \left(\sum_{j=1}^n a_{ij} x_j \right) = \mathbf{x}^T \mathbf{A} \mathbf{x} \quad (1.15)$$

Since $\mathbf{A} \mathbf{x}$ represents a vector, the triple product of Eq. (1.15) will be also written as a dot product:

$$c = \mathbf{x}^T \mathbf{A} \mathbf{x} = (\mathbf{x} \cdot \mathbf{A} \mathbf{x}) \quad (1.16)$$

1.5.5 Functions

Just as a function of a single variable is represented as $f(x)$, a function of n independent variables x_1, x_2, \dots, x_n is written as

$$f(\mathbf{x}) = f(x_1, x_2, \dots, x_n) \quad (1.17)$$

We will deal with many functions of vector variables. To distinguish between functions, subscripts will be used. Thus, the i th function is written as

$$g_i(\mathbf{x}) = g_i(x_1, x_2, \dots, x_n) \quad (1.18)$$

If there are m functions $g_i(\mathbf{x})$; $i = 1$ to m , these will be represented in the vector form

$$\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}) \ g_2(\mathbf{x}) \ \dots \ g_m(\mathbf{x})]^T \quad (1.19)$$

Throughout the text it is *assumed* that all functions are *continuous* and at least *twice continuously differentiable*. A function $f(\mathbf{x})$ of n variables is called *continuous* at a point \mathbf{x}^* if for any $\varepsilon > 0$, there is a $\delta > 0$ such that

$$|f(\mathbf{x}) - f(\mathbf{x}^*)| < \varepsilon \quad (1.20)$$

whenever $\|\mathbf{x} - \mathbf{x}^*\| < \delta$. Thus, for all points \mathbf{x} in a small neighborhood of the point \mathbf{x}^* , a change in the function value from \mathbf{x}^* to \mathbf{x} is small when the function is continuous. A continuous function need not be differentiable. *Twice-continuous differentiability* of a function implies that it is not only differentiable two times but also that its second derivative is continuous. Figures 1.5(A) and 1.5(B) show continuous functions. The function shown in Fig. 1.5(A) is differentiable everywhere, whereas the function of Fig. 1.5(B) is not differentiable at points x_1 , x_2 , and x_3 . Figure 1-5(C) provides an example in which f is not a function because it has infinite values at x_1 . Figure 1-5(D) provides an example of a discontinuous function. As examples, $f(x) = x^3$ and $f(x) = \sin x$ are continuous functions everywhere, and they are also continuously differentiable. However, the function $f(x) = |x|$ is continuous everywhere but not differentiable at $x = 0$.

1.5.6 U.S.-British versus SI Units

The design problem formulation and the methods of optimization do not depend on the units of measure used. Thus, it does not matter which units are used in defining the problem. However, the final form of some of the analytical expressions for the problem does depend on the units used. In the text, we shall use both U.S.-British and SI units in examples and exercises. Readers unfamiliar with either system of units should not feel at a disadvantage when reading and understanding the material. It is simple to switch from one system of units to the other. To facilitate the conversion from U.S.-British to SI units or vice versa, Table 1-1 gives conversion factors for the most commonly used quantities. For a complete list of the conversion factors, the ASTM (1980) publication can be consulted.

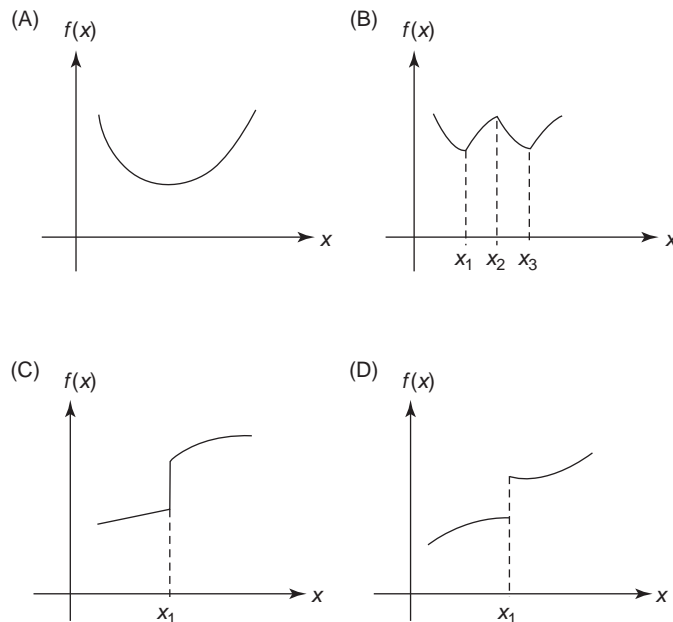


FIGURE 1-5 Continuous and discontinuous functions. (A) Continuous function. (B) Continuous function. (C) Not a function. (D) Discontinuous function.

TABLE 1-1 Conversion Factors Between U.S.-British and SI Units

<i>To convert from U.S.-British</i>	<i>To SI units</i>	<i>Multiply by</i>
Acceleration		
foot/second ² (ft/s ²)	meter/second ² (m/s ²)	0.3048*
inch/second ² (in/s ²)	meter/second ² (m/s ²)	0.0254*
Area		
foot ² (ft ²)	meter ² (m ²)	0.09290304*
inch ² (in ²)	meter ² (m ²)	6.4516 E-04*
Bending moment or torque		
pound force inch (lbf-in)	Newton meter (N·m)	0.1129848
pound force foot (lbf-ft)	Newton meter (N·m)	1.355818
Density		
pound mass/inch ³ (lbm/in ³)	kilogram/meter ³ (kg/m ³)	27,679.90
pound mass/foot ³ (lbm/ft ³)	kilogram/meter ³ (kg/m ³)	16.01846
Energy or Work		
British thermal unit (BTU)	Joule (J)	1055.056
foot-pound force (ft-lbf)	Joule (J)	1.355818
kilowatt-hour (KWh)	Joule (J)	3,600,000*
Force		
kip (1000 lbf)	Newton (N)	4448.222
pound force (lbf)	Newton (N)	4.448222
Length		
foot (ft)	meter (m)	0.3048*
inch (in)	meter (m)	0.0254*
mile (mi), U.S. statute	meter (m)	1609.347
mile (mi), International, nautical	meter (m)	1852*
Mass		
pound mass (lbm)	kilogram (kg)	0.4535924
slug (lbf·s ² /ft)	kilogram (kg)	14.5939
ton (short, 2000 lbm)	kilogram (kg)	907.1847
ton (long, 2240 lbm)	kilogram (kg)	1016.047
tonne (t, metric ton)	kilogram (kg)	1000*
Power		
foot-pound/minute (ft-lbf/min)	Watt (W)	0.02259697
horsepower (550 ft-lbf/s)	Watt (W)	745.6999
Pressure or stress		
atmosphere (std) (14.7 lbf/in ²)	Newton/meter ² (N/m ² or Pa)	101,325*
one bar (b)	Newton/meter ² (N/m ² or Pa)	100,000*
pound/foot ² (lbf/ft ²)	Newton/meter ² (N/m ² or Pa)	47.88026
pound/inch ² (lbf/in ² or psi)	Newton/meter ² (N/m ² or Pa)	6894.757
Velocity		
foot/minute (ft/min)	meter/second (m/s)	0.00508*
foot/second (ft/s)	meter/second (m/s)	0.3048*
knot (nautical mi/h), international	meter/second (m/s)	0.5144444
mile/hour (mi/h), international	meter/second (m/s)	0.44704*
mile/hour (mi/h), international	kilometer/hour (km/h)	1.609344*
mile/second (mi/s), international	kilometer/second (km/s)	1.609344*

TABLE 1-1 *continued*

<i>To convert from U.S.-British</i>	<i>To SI units</i>	<i>Multiply by</i>
Volume		
foot ³ (ft ³)	meter ³ (m ³)	0.02831685
inch ³ (in ³)	meter ³ (m ³)	1.638706 E-05
gallon (Canadian liquid)	meter ³ (m ³)	0.004546090
gallon (U.K. liquid)	meter ³ (m ³)	0.004546092
gallon (U.S. dry)	meter ³ (m ³)	0.004404884
gallon (U.S. liquid)	meter ³ (m ³)	0.003785412
one liter (L)	meter ³ (m ³)	0.001*
ounce (U.K. fluid)	meter ³ (m ³)	2.841307 E-05
ounce (U.S. fluid)	meter ³ (m ³)	2.957353 E-05
pint (U.S. dry)	meter ³ (m ³)	5.506105 E-04
pint (U.S. liquid)	meter ³ (m ³)	4.731765 E-04
quart (U.S. dry)	meter ³ (m ³)	0.001101221
quart (U.S. liquid)	meter ³ (m ³)	9.463529 E-04

* An asterisk indicates the exact conversion factor.

2 Optimum Design Problem Formulation

Upon completion of this chapter, you will be able to:

- Translate a descriptive statement of the design problem into a mathematical statement for optimization using a five-step process
- Identify and define the problem's design variables
- Identify a function for the problem that needs to be optimized
- Identify and define the problem's constraints

It is generally accepted that the *proper definition and formulation of a problem* takes roughly 50 percent of the total effort needed to solve it. Therefore, it is critical to follow well defined procedures for formulating design optimization problems. It is generally assumed in this text that *various preliminary analyses have been completed and a detailed design of a concept or a subproblem needs to be carried out*. Students should bear in mind that a considerable number of analyses usually have to be performed before reaching this stage of design optimization. In this chapter, we describe the process of transforming the design of a selected system/subsystem into an optimum design problem using several simple and moderately complex applications. More advanced applications are discussed in later chapters.

The *importance of properly formulating* a design optimization problem must be stressed because the optimum solution will only be as good as the formulation. For example, if we forget to include a critical constraint in the formulation, the optimum solution will most likely violate it because optimization methods tend to exploit deficiencies in design models. Also, if we have too many constraints or if they are inconsistent, there may not be a solution to the design problem. However, once the problem is properly formulated, good software is usually available to solve it. For most design optimization problems, we shall use the following *five-step* formulation procedure:

- Step 1: Project/problem statement.
- Step 2: Data and information collection.
- Step 3: Identification/definition of design variables.
- Step 4: Identification of a criterion to be optimized.
- Step 5: Identification of constraints.

2.1 The Problem Formulation Process

The formulation of an optimum design problem involves translating a descriptive statement of the problem into a well defined mathematical statement. We shall describe the tasks to be performed in each of the five steps to develop a mathematical formulation for the design optimization problem. These steps are illustrated with several examples in subsequent sections of this chapter.

Are the
project
goals
clear?

2.1.1 Step 1: Project/Problem Statement

The formulation process begins by developing a descriptive statement for the project/problem, which is usually done by the project's owner/sponsor. The statement describes the overall *objectives* of the project and the *requirements* to be met.

Is all the
information
available to
solve the
problem?

2.1.2 Step 2: Data and Information Collection

To develop a mathematical formulation of the problem, we need to gather material properties, performance requirements, resource limits, cost of raw materials, and other relevant information. In addition, most problems require the capability to *analyze trial designs*. Therefore, *analysis procedures* and *analysis tools* must be identified at this stage. In many cases, the project statement is vague, and assumptions about the problem need to be made in order to formulate and solve it. Some of the design data and expressions may depend on design variables that are identified in the next step. Therefore, such information will need to be specified later in the formulation process.

What are
these
variables?
How do I
identify
them?

2.1.3 Step 3: Identification/Definition of Design Variables

The next step in the formulation process is to identify a set of variables that describe the system, called *design variables*. In general, they are referred to as optimization variables and are regarded as *free* because we should be able to assign any value to them. Different values for the variables produce different designs. The design variables should be independent of each other as far as possible. If they are dependent, then their values cannot be specified independently. The number of independent design variables specifies the *design degrees of freedom* for the problem.

For some problems, different sets of variables can be identified to describe the same system. The problem formulation will depend on the selected set. Once the design variables are given numerical values, we have a *design of the system*. Whether this design *satisfies all requirements* is another question. We shall introduce a number of concepts to investigate such questions in later chapters.

If proper design variables are not selected for a problem, the formulation will be either incorrect or not possible at all. At the initial stage of problem formulation, all options of identifying design variables should be investigated. Sometimes it may be desirable to designate more design variables than apparent design degrees of freedom. This gives an added flexibility in the problem formulation. Later, it is possible to assign a fixed numerical value to any variable and thus eliminate it from the problem formulation.

At times it is difficult to identify clearly a problem's design variables. In such a case, a complete list of all variables may be prepared. Then, by considering each variable individually, we can decide whether it is an optimization variable. If it is a valid design variable, then the designer should be able to specify a numerical value for it to select a trial design.

We shall use the term "*design variables*" to indicate all unknowns of the optimization problem and represent in the vector \mathbf{x} . To summarize, the following considerations should be given in identifying design variables for a problem.

- Design variables should be independent of each other as far as possible. If they are not, then there must be some equality constraints between them (explained later). Conversely, if there are equality constraints in the problem, then the design variables are dependent.
- A minimum number of design variables required to formulate a design optimization problem properly exists.
- As many independent parameters as possible should be designated as design variables at the problem formulation phase. Later on, some of the variables can be assigned fixed values.
- A numerical value should be given to each variable once design variables have been defined to determine if a trial design of the system is specified.

2.1.4 Step 4: Identification of a Criterion to Be Optimized

How do I know that my design is the best?

There can be many feasible designs for a system, and some are better than others. To compare different designs, we must have a criterion. The criterion must be a scalar function whose numerical value can be obtained once a design is specified, i.e., it must be a *function of the design variable vector* \mathbf{x} . Such a criterion is usually called an *objective function* for the optimum design problem, which needs to be *maximized* or *minimized* depending on problem requirements. A criterion that is to be minimized is usually called the *cost function* in engineering literature, which is the term used throughout this text. It is emphasized that a *valid objective function must be influenced directly or indirectly by the variables of the design problem*; otherwise, it is not a meaningful objective function. Note that an optimized design has the *best* value for the objective function.

The selection of a proper objective function is an important decision in the design process. Some examples of objective functions include: cost (to be minimized), profit (to be maximized), weight (to be minimized), energy expenditure (to be minimized), ride quality of a vehicle (to be maximized), and so on. In many situations an obvious function can be identified, e.g., we always want to minimize the cost of manufacturing goods or maximize return on an investment. In some situations, two or more objective functions may be identified. For example, we may want to minimize the weight of a structure and at the same time minimize the deflection or stress at a certain point. These are called *multiobjective design optimization* problems, and they are discussed in a later chapter.

For some design problems, it is not obvious what the objective function should be or how it should relate to the design variables. Some insight and experience may be needed to identify a proper objective function. For example, consider the optimization of a passenger car. What are the design variables for the car? What is the objective function, and what is its functional form in terms of design variables? Although this is a very practical problem, it is quite complex. Usually, such problems are divided into several smaller subproblems and each one is formulated as an optimum design problem. The design of the passenger car for a given capacity and for certain performance specifications can be divided into a number of such subproblems: optimization of the trunk lid, doors, side panels, roof, seats, suspension system, transmission system, chassis, hood, power plant, bumpers, and so on. Each subproblem is now manageable and can be formulated as an optimum design problem.

2.1.5 Step 5: Identification of Constraints

What restrictions do I have on my design?

All restrictions placed on a design are collectively called *constraints*. The final step in the formulation process is to identify all constraints and develop expressions for them. Most realistic systems must be designed and fabricated within given *resources* and *performance requirements*. For example, structural members should not fail under normal operating loads. Vibration frequencies of a structure must be different from the operating frequency of the machine it supports; otherwise, resonance can occur causing catastrophic failure. Members

must fit into available amounts of space. All these and other constraints must depend on the design variables, since only then do their values change with different trial designs; i.e., a meaningful constraint must be a function of at least one design variable. Several concepts and terms related to constraints are explained in the following paragraphs.

Linear and Nonlinear Constraints Many constraint functions have only first-order terms in design variables. These are called *linear constraints*. *Linear programming problems* have only linear constraint and objective functions. More general problems have nonlinear cost and/or constraint functions. These are called *nonlinear programming problems*. Methods to treat both linear and nonlinear constraint and objective functions have been developed in the literature.

Feasible Design The design of a system is a set of numerical values assigned to the design variables (i.e., a particular design variable vector \mathbf{x}). Even if this design is absurd (e.g., negative radius) or inadequate in terms of its function, it can still be called a design. Clearly, some designs are useful and others are not. A design meeting all requirements is called a *feasible design* (*acceptable* or *workable*). An *infeasible design* (*unacceptable*) does not meet one or more of the requirements.

Equality and Inequality Constraints Design problems may have equality as well as inequality constraints. The problem statement should be studied carefully to determine which requirements need to be formulated as equalities and which ones as inequalities. For example, a machine component may be required to move precisely by Δ to perform the desired operation, so we must treat this as an equality constraint. A feasible design must satisfy precisely all equality constraints. Also, most design problems have inequality constraints. Inequality constraints are also called *unilateral constraints* or *one-sided constraints*. Note that the *feasible region* with respect to an inequality constraint is much larger than the same constraint expressed as an equality. To illustrate the difference between equality and inequality constraints, we consider a constraint written in both equality and inequality forms. Figure 2-1(A) shows the equality constraint $x_1 = x_2$. Feasible designs with respect to the constraint must lie on the straight line A–B. However, if the constraint is written as an inequality $x_1 \leq x_2$, the feasible region is much larger, as shown in Fig. 2-1(B). Any point on the line A–B or above it gives a feasible design.

Implicit Constraints Some constraints are quite simple, such as the smallest and largest allowable values of the design variables, whereas more complex ones may be indirectly influenced by design variables. For example, deflection at a point in a large structure depends on its design. However, it is impossible to express deflection as an explicit function of the design variables except for very simple structures. These are called *implicit constraints*. When there are implicit functions in the problem formulation, it is not possible to formulate the problem functions explicitly in terms of design variables alone. Instead, we must use some intermediate variables in the problem formulation. We shall discuss formulations having implicit functions in Chapter 14.

2.2 Design of a Can

Step 1: Project/Problem Statement The purpose of this project is to design a can to hold at least 400 ml of liquid, as well as to meet other design requirements ($1 \text{ ml} = 1 \text{ cm}^3$). The cans will be produced in the billions so it is desirable to minimize manufacturing costs. Since cost can be directly related to the surface area of the sheet metal, it is reasonable to

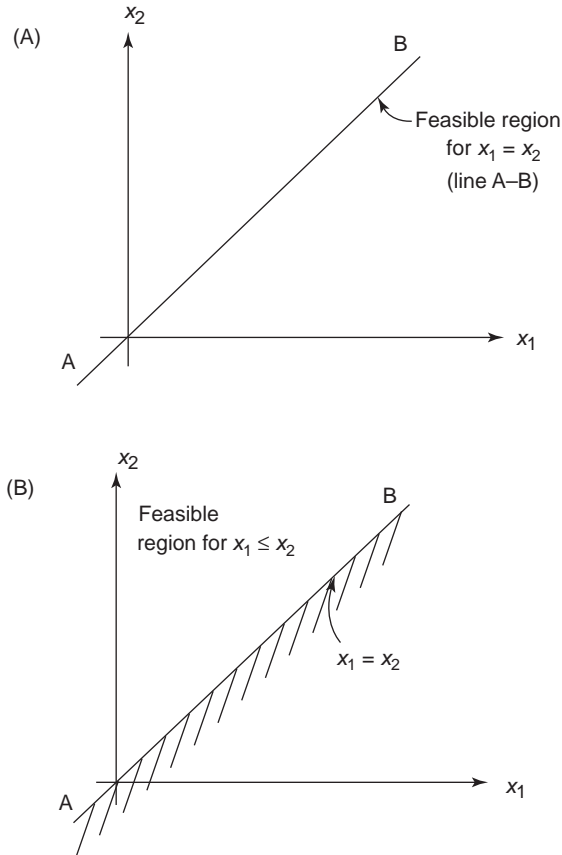


FIGURE 2-1 Distinction between equality and inequality constraints. (A) Feasible region for constraint $x_1 = x_2$ (line A–B). (B) Feasible region for constraint $x_1 \leq x_2$ (line A–B and the region above it).

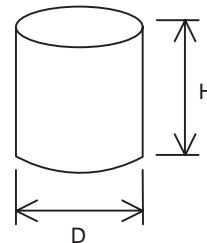
minimize the amount of sheet metal required to fabricate the can. Fabrication, handling, aesthetics, and shipping considerations impose the following restrictions on the size of the can: the diameter should be no more than 8 cm and no less than 3.5 cm, whereas the height should be no more than 18 cm and no less than 8 cm.

Step 2: Data and Information Collection Given in the project statement.

Step 3: Identification/Definition of Design Variables The two design variables are defined as

D = diameter of the can, cm

H = height of the can, cm



Step 4: Identification of a Criterion to Be Optimized The design objective is to minimize the total surface area S of the sheet metal for the three parts of the cylindrical can: the

surface area of the cylinder (circumference \times height) and the surface area of the two ends. Therefore, the optimization criterion or *cost function* (the total area of sheet metal), is written as

$$S = \pi DH + \frac{\pi}{2} D^2, \text{ cm}^2 \quad (\text{a})$$

Step 5: Identification of Constraints The first constraint is that the can must hold *at least* 400 cm^3 of fluid, which is written as

$$\frac{\pi}{4} D^2 H \geq 400, \text{ cm}^3 \quad (\text{b})$$

If it had been stated that the “can must hold 400 ml of fluid,” then the preceding volume constraint would be an equality. The other constraints on the size of the can are:

$$\begin{aligned} 3.5 \leq D \leq 8, \text{ cm} \\ 8 \leq H \leq 18, \text{ cm} \end{aligned} \quad (\text{c})$$

The explicit constraints on design variables have many different names in the literature, such as the *side constraints*, *technological constraints*, *simple bounds*, *sizing constraints*, and *upper and lower limits on the design variables*. Note that for the present problem there are really four constraints in Eq. (c). Thus, the problem has two design variables and a total of five inequality constraints. Note also that the cost function and the first constraint are non-linear in variables, whereas the remaining constraints are linear.

2.3 Insulated Spherical Tank Design

Step 1: Project/Problem Statement The goal of this project is to choose insulation thickness t to minimize the life-cycle cooling cost for a spherical tank. The cooling costs include the cost of installing and running the refrigeration equipment, and the cost of installing the insulation. Assume a 10-year life, 10 percent annual interest rate, and no salvage value. The tank has already been designed having r (m) as its radius.

Step 2: Data and Information Collection To formulate this design optimization problem, we need some data and expressions. To calculate the volume of the insulation material, we require the surface area of the spherical tank, which is given as

$$A = 4\pi r^2, \text{ m}^2 \quad (\text{a})$$

To calculate the capacity of the refrigeration equipment and the cost of its operation, we need to calculate the annual heat gain G , which is given as

$$G = \frac{(365)(24)(\Delta T)A}{c_1 t}, \text{ Watt-hours} \quad (\text{b})$$

where ΔT is the average difference between the internal and external temperatures in Kelvin, c_1 is the thermal resistivity per unit thickness in Kelvin-meter per Watt, and t is the insula-

tion thickness in meters. ΔT can be estimated from the historical data for temperatures in the region in which the tank is to be used. Let c_2 = the insulation cost per cubic meter ($\$/\text{m}^3$), c_3 = the cost of the refrigeration equipment per Watt-hour of capacity ($\$/\text{Wh}$), and c_4 = the annual cost of running the refrigeration equipment per Watt-hour ($\$/\text{Wh}$).

Step 3: Identification/Definition of Design Variables Only one design variable is identified for this problem:

$$t = \text{the insulation thickness, m}$$

Step 4: Identification of a Criterion to be Optimized The goal is to minimize the life-cycle cooling cost of refrigeration for the spherical tank over 10 years. The life-cycle cost has three components: cost of insulation, cost of refrigeration equipment, and cost of operations for 10 years. Once the annual operations cost has been converted to the present cost, the total cost is given as

$$\text{Cost} = c_2At + c_3G + c_4G[\text{uspwf}(0.1, 10)]; \quad [\text{uspwf}(0.1, 10)] = 6.14457 \quad (\text{c})$$

where *uspwf* is the uniform series present worth factor (see Appendix A). Note that to calculate the volume of the insulation as At , it is assumed that the insulation thickness is much smaller than the radius of the spherical tank; i.e., $t \ll r$.

Step 5: Identification of Constraints Although no constraints are indicated in the problem statement, it is important to require that the insulation thickness be nonnegative; i.e., $t \geq 0$. Although this may appear to be an obvious requirement that need not be included in the mathematical formulation of the problem, it is important to include the constraint in the formulation. Without its explicit inclusion, the mathematics of optimization may assign negative values to thickness which is, of course, meaningless. Note also that in reality t cannot be zero because it appears in the denominator of the expression for G . Therefore, the constraint should really be expressed as $t > 0$. However, *strict inequalities* cannot be treated mathematically or numerically in the solution process. We must allow the possibility of satisfying inequalities as equalities; i.e., we must allow the possibility that $t = 0$ in the solution process. Therefore, a more realistic constraint is $t \geq t_{\min}$, where t_{\min} is the smallest insulation thickness available in the market.

EXAMPLE: Formulation with Intermediate Variables

Summary of the problem formulation for the design optimization of insulation for a spherical tank formulation with intermediate variables is as follows:

Specified data: $r, \Delta T, c_1, c_2, c_3, c_4, t_{\min}$

Design variable: t, m

Intermediate variables:

$$A = 4\pi r^2$$

$$G = \frac{(365)(24)(\Delta T)A}{c_1 t}$$

Cost function: Minimize the life-cycle cooling cost of refrigeration of the spherical tank

$$Cost = c_2At + c_3G + 6.14457c_4G$$

Constraint:

$$t \geq t_{\min}$$

Note that A and G may also be treated as design variables in this formulation. In such a case, A must be assigned a fixed numerical value and the expression for G must be treated as an equality constraint.

EXAMPLE: Formulation with Design Variables Only

Summary of the problem formulation for the design optimization of insulation for a spherical tank formulation in terms of the design variable only is as follows:

Specified data: $r, \Delta T, c_1, c_2, c_3, c_4, t_{\min}$

Design variable: t, m

Cost function: Minimize the life-cycle cooling cost of refrigeration of the spherical tank

$$Cost = at + \frac{b}{t}$$

$$a = 4c_2\pi r^2, \quad b = \frac{(c_3 + 6.14457c_4)}{c_1}(365)(24)(\Delta T)(4\pi r^2)$$

Constraint:

$$t \geq t_{\min}$$

2.4 Saw Mill Operation

Step 1: Project/Problem Statement A company owns two saw mills and two forests. Table 2-1 shows the capacity of each mill (logs/day) and the distances between forests and mills (km). Each forest can yield up to 200 logs/day for the duration of the project, and the cost to transport the logs is estimated at \$0.15/km/log. At least 300 logs are needed each day. The goal is to minimize the total cost of transportation of logs each day.

Step 2: Data and Information Collection Data are given in Table 2-1 and in the problem statement.

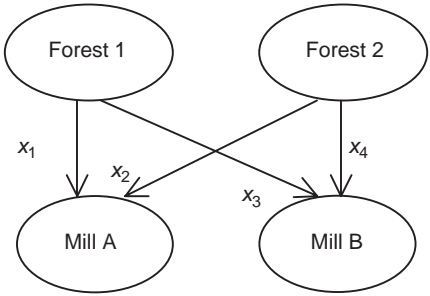
Step 3: Identification/Definition of Design Variables The design problem is to determine how many logs to ship from Forest i to Mill j . Therefore, the design variables for the problem are identified and defined as follows:

x_1 = number of logs shipped from Forest 1 to Mill A

x_2 = number of logs shipped from Forest 2 to Mill A

TABLE 2-1 Data for Saw Mill Operation

Mill	Distance, km		Mill capacity/day
	Forest 1	Forest 2	
A	24.0	20.5	240 logs
B	17.2	18.0	300 logs



x_3 = number of logs shipped from Forest 1 to Mill B
 x_4 = number of logs shipped from Forest 2 to Mill B

Note that if we assign numerical values to these variables, an operational plan for the project is specified and the cost of transportation of logs per day can be calculated. The selected design may or may not satisfy all other requirements.

Step 4: Identification of a Criterion to Be Optimized The design objective is to minimize the daily cost of transporting the logs to the mills. The cost of transportation, which depends on the distance between the forests and the mills, is:

$$\begin{aligned} \text{Cost} &= 24(0.15)x_1 + 20.5(0.15)x_2 + 17.2(0.15)x_3 + 18(0.15)x_4 \\ &= 3.600x_1 + 3.075x_2 + 2.580x_3 + 2.700x_4, \$ \end{aligned} \quad (a)$$

Step 5: Identification of Constraints The constraints for the problem are based on the capacity of the mills and the yield of the forests:

$$\begin{aligned} x_1 + x_2 &\leq 240 && \text{(Mill A capacity)} \\ x_3 + x_4 &\leq 300 && \text{(Mill B capacity)} \\ x_1 + x_3 &\leq 200 && \text{(Forest 1 yield)} \\ x_2 + x_4 &\leq 200 && \text{(Forest 2 yield)} \end{aligned} \quad (b)$$

The constraint on the number of logs needed for each day is expressed as

$$x_1 + x_2 + x_3 + x_4 \geq 300 \quad \text{(demand for logs)} \quad (c)$$

For a realistic problem formulation, all design variables must be nonnegative, i.e.,

$$x_i \geq 0; i = 1 \text{ to } 4 \quad (d)$$

The problem has four design variables, five inequality constraints, and four nonnegativity constraints on the variables. Note that all functions of the problem are linear in design variables, so it is a *linear programming problem*. Note also that for a meaningful solution, all design variables must have *integer* values. Such problems are called *integer programming problems*, which require special solution methods. Some such methods are discussed in Chapter 15.

2.5 Design of a Two-Bar Bracket

Step 1: Project/Problem Statement The objective of this project is to design a two-bar bracket (shown in Fig. 2-2) to support a force W without structural failure. The force is applied at an angle θ , which is between 0 and 90° ; h is the height, and s is the base width for the bracket. The brackets will be produced in large quantities. It has also been determined that the total cost of the bracket (material, fabrication, maintenance, and so on) is directly related to the size of the two bars. Thus, the design objective is to minimize the total mass of the bracket while satisfying performance, fabrication, and space limitations.

Step 2: Data and Information Collection Some data and information are needed to formulate the problem. First, the force W and its angle of application θ need to be specified. Since the bracket may be used in several applications, it may not be possible to specify just one angle for W . It is possible to formulate the design optimization problem such that a range is specified for angle θ ; i.e., the force W may be applied at any angle within that specified range. In this case, the formulation will be slightly more complex because performance requirements will need to be satisfied for each angle of application. In the present formulation, it is assumed that the angle θ is specified. Secondly, the material to be used for the bars must be specified because the material properties are needed to formulate the optimization criterion and performance requirements. Whether the two bars are to be fabricated using the same material also needs to be determined. In the present formulation, it is assumed that they are, although it may be prudent to assume otherwise for some advanced applications. In addition, we need to determine the fabrication and space limitations for the bracket, e.g., limitations on the size of the bars, height, and base width.

In formulating the design problem, we also need to define *structural performance* more precisely. Forces F_1 and F_2 carried by bars 1 and 2, respectively, can be used to define failure

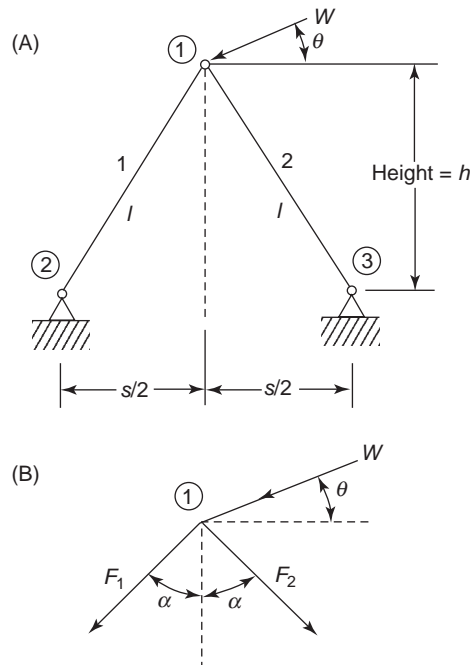


FIGURE 2-2 The two-bar bracket. (A) The structure. (B) Free body diagram for node 1.

conditions for the bars. To compute these forces, we use the principle of *static equilibrium*. Using the *free-body diagram* for Node 1 (shown in Fig. 2-2), equilibrium of forces in the horizontal and vertical directions gives:

$$\begin{aligned} -F_1 \sin \alpha + F_2 \sin \alpha &= W \cos \theta \\ -F_1 \cos \alpha - F_2 \cos \alpha &= W \sin \theta \end{aligned} \quad (a)$$

From the geometry of Fig. 2-2, $\sin \alpha = 0.5s/l$ and $\cos \alpha = h/l$, where l is the length of members given as $l = \sqrt{h^2 + (0.5s)^2}$. Note that F_1 and F_2 are shown as tensile forces in the free-body diagram. The solution to Eq. (a) will determine the magnitude and direction of the forces. In addition, the *tensile force will be taken as positive*. Thus, the bar will be in compression if the force carried by it has negative value. By solving the two equations simultaneously for the unknowns F_1 and F_2 , we obtain

$$\begin{aligned} F_1 &= -0.5Wl \left[\frac{\sin \theta}{h} + \frac{2 \cos \theta}{s} \right] \\ F_2 &= -0.5Wl \left[\frac{\sin \theta}{h} - \frac{2 \cos \theta}{s} \right] \end{aligned} \quad (b)$$

To avoid bar failure due to overstressing, we need to calculate bar stress. If we know the force carried by a bar, then the stress σ can be calculated as force divided by the bar's cross-sectional area (stress = force/area). The SI unit for stress is Newton/m² (N/m²), also called Pascal (Pa), whereas the U.S.-British unit is pound/in.² (written as psi). The expression for the cross-sectional area depends on the cross-sectional shape used for the bars and selected design variables. Therefore, a structural shape for the bars and associated design variables must be selected. This is illustrated later in the formulation process.

In addition to analysis equations, we need to define the properties of the selected material. Several formulations for optimum design of the bracket are possible depending on the requirements of the application. To illustrate, a material with known properties is assumed for the bracket. However, the structure can be optimized using other materials along with their associated fabrication costs. Solutions can then be compared to select the best possible one for the structure. For the selected material, let ρ be the mass density and σ_a be the allowable design stress, which is taken as a positive quantity. As a performance requirement, it is assumed that if the stress exceeds this allowable value, the bar is considered to have failed. The *allowable stress* is defined as the material failure stress (a property of the material) divided by a factor of safety greater than one. We may also call it the *design stress*. In addition, it is assumed that the allowable stress is calculated in such a way that the buckling failure of a bar in compression is avoided.

Step 3: Identification/Definition of Design Variables Several sets of design variables may be identified for the two-bar structure. The height h and span s can be treated as design variables in the initial formulation. Later, they may be assigned numerical values, if desired, to eliminate them from the formulation. Other design variables will depend on the cross-sectional shape of bars 1 and 2. Several cross-sectional shapes are possible, as shown in Fig. 2-3, where design variables for each shape are also identified. Note that for many cross-sectional shapes, different design variables can be selected. For example, in the case of the circular tube in Fig. 2-3(A), the outer diameter d_o and the ratio between the inner and outer diameters $r = d_i/d_o$ may be selected as the design variables. Or, d_o and d_i may be selected as design variables. However, it is not desirable to designate d_o , d_i , and r as the design variables because they are not independent of each other. Similar remarks can be made for the design variables associated with other cross sections, as shown in Fig. 2-3.

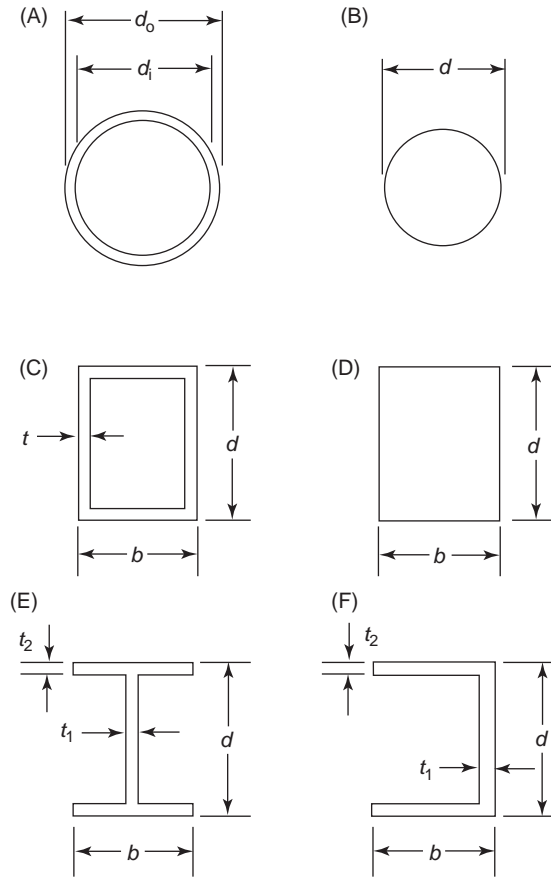


FIGURE 2-3 Bar cross-sectional shapes for two-bar structure. (A) Circular tube. (B) Solid circular. (C) Rectangular tube. (D) Solid rectangular. (E) I-section. (F) Channel section.

As an example of problem formulation, consider the design of a bracket with hollow circular tubes, as shown in Fig. 2-3(A). The inner and outer diameters d_i and d_o and wall thickness t may be identified as the design variables, although they are not all independent of each other. We cannot specify $d_i = 10$, $d_o = 12$, and $t = 2$ because it violates the physical condition $t = 0.5(d_o - d_i)$. Therefore, if we formulate the problem with d_i , d_o , and t as design variables, we must also impose the constraint $t = 0.5(d_o - d_i)$. This type of formulation is usually unnecessary because we could substitute for t in all equations to eliminate it from the problem, thus reducing the number of design variables and constraints. To illustrate a formulation of the problem, let the design variables be defined as

- x_1 = height h of the bracket
- x_2 = span s of the bracket
- x_3 = outer diameter of bar 1
- x_4 = inner diameter of bar 1
- x_5 = outer diameter of bar 2
- x_6 = inner diameter of bar 2

In terms of these variables, the cross-sectional areas A_1 and A_2 of bars 1 and 2 are given as

$$A_1 = \frac{\pi}{4}(x_3^2 - x_4^2); \quad A_2 = \frac{\pi}{4}(x_5^2 - x_6^2) \quad (c)$$

Once the problem is formulated in terms of the selected six design variables, it is always possible to modify it to meet more specialized needs. For example, the height x_1 may be assigned a fixed numerical value, thus eliminating it from the problem formulation. In addition, complete symmetry of the structure may be required to make its fabrication easier; i.e., it may be necessary for the two bars to have the same cross section, size, and material. In such a case, we set $x_3 = x_5$ and $x_4 = x_6$ in all expressions of the problem formulation. Such modifications are left as exercises.

Step 4: Identification of a Criterion to Be Optimized The structure's mass is identified as the objective function in the problem statement. Since it is to be minimized, it is called the *cost function* for the problem. An expression for the mass is determined by the cross-sectional shape of the bars and associated design variables. For the hollow circular tubes and selected design variables, the total mass of the structure is calculated as (density \times material volume):

$$Mass = \rho[l(A_1 + A_2)] = \left[\rho \sqrt{x_1^2 + (0.5x_2)^2} \right] \frac{\pi}{4}(x_3^2 - x_4^2 + x_5^2 - x_6^2) \quad (d)$$

Note that if the outer diameter and the ratio between the inner and outer diameter are selected as design variables, the form of the mass function changes. Thus, the *final form* depends on design variables selected for the problem. Expressions for the mass of the other cross-sectional shapes shown in Fig. 2-3 can be easily written, if desired.

Step 5: Identification of Constraints It is important to include all constraints in the problem formulation because the final solution depends on them. For the two-bar structure, the constraints are on the stress in the bars and the design variables themselves. These constraints will be formulated for hollow circular tubes using the previously defined design variables. They can be similarly formulated for other sets of design variables and cross-sectional shapes.

To avoid over-stressing a bar, the calculated stress σ must not exceed the material allowable stress $\sigma_a > 0$. The stresses σ_1 and σ_2 in the two bars are calculated as force/area:

$$\begin{aligned} \sigma_1 &= \frac{F_1}{A_1} \text{ (stress in bar 1)} \\ \sigma_2 &= \frac{F_2}{A_2} \text{ (stress in bar 2)} \end{aligned} \quad (e)$$

Note that to treat positive and negative stresses (tension and compression), we must use the absolute value of the calculated stress in writing the constraints. From Eq. (b), F_1 (the force in bar 1) is always negative, so it is a compressive force. If $(\sin \theta)/x_1 \geq (2\cos \theta)/x_2$ in Eq. (b) (where $x_1 = h$ and $x_2 = s$ have been used), then F_2 is also a compressive force and is thus negative. Therefore, the stress constraints for bars 1 and 2 are given as

$$\begin{aligned} -\sigma_1 &\leq \sigma_a \quad \text{(stress constraint for bar 1)} \\ -\sigma_2 &\leq \sigma_a \quad \text{(stress constraint for bar 2)} \end{aligned} \quad (f)$$

If $(\sin \theta)/x_1 < (2\cos \theta)/x_2$ in Eq. (b), then F_2 becomes a tensile force (thus positive) and the stress constraint for bar 2 becomes

$$\sigma_2 \leq \sigma_a \quad (\text{stress constraint for bar 2 when in tension}) \quad (g)$$

Both compressive and tensile stress constraints for bar 2 can be included in the problem formulation. The solution methods described in subsequent chapters will automatically select the appropriate constraint among compressive or tensile choices.

Finally, to impose fabrication and space limitations, constraints on the design variables are imposed as

$$x_{il} \leq x_i \leq x_{iu}; i = 1 \text{ to } 6 \quad (h)$$

where x_{il} and x_{iu} are the minimum and maximum allowed values for the i th design variable. Their numerical values must be specified before the problem can be solved.

Note that the expression for bar stress changes if different design variables are chosen for circular tubes, or if a different cross-sectional shape is chosen for the bars. For example, inner and outer radii, mean radius and wall thickness, or outside diameter and the ratio of inside to outside diameter as design variables will all produce different expressions for the cross-sectional areas and stresses. *These results show that the choice of design variables greatly influences the problem formulation.*

Note also that we had to first *analyze* the structure (calculate its response to given inputs) to write the constraints properly. It was only after we had calculated the forces in the bars that we were able to write the constraints. This is an important step in any engineering design problem formulation: *We must be able to analyze the system before we can formulate the design optimization problem.*

In the following examples, we summarize two formulations of the problem. The first formulation uses several intermediate variables, which is useful when the problem is transcribed into a computer program. Because this formulation involves smaller expressions of various quantities, it is easier to write and debug a computer program. In the second formulation, all intermediate variables are eliminated to obtain the formulation exclusively in terms of design variables. This formulation has slightly more complex expressions. It is important to note that the second formulation may not be possible for all applications because some problem functions may only be implicit functions of the design variables. One such formulation is presented in Chapter 14.

EXAMPLE: Formulation with Intermediate Variables

Summary of the problem formulation for optimum design of the two-bar bracket using intermediate variables is as follows:

Specified data: $W, \theta, \sigma_a > 0, x_{il}$ and $x_{iu}, i = 1$ to 6

Design variables: $x_1, x_2, x_3, x_4, x_5, x_6$

Intermediate variables:

$$\text{Bar cross-sectional areas: } A_1 = \frac{\pi}{4}(x_3^2 - x_4^2); \quad A_2 = \frac{\pi}{4}(x_5^2 - x_6^2)$$

$$\text{Length of bars: } l = \sqrt{x_1^2 + (0.5x_2)^2}$$

$$F_1 = -0.5Wl \left[\frac{\sin \theta}{x_1} + \frac{2 \cos \theta}{x_2} \right]$$

Forces in bars:

$$F_2 = -0.5Wl \left[\frac{\sin \theta}{x_1} - \frac{2 \cos \theta}{x_2} \right]$$

$$\text{Bar stresses: } \sigma_1 = \frac{F_1}{A_1}; \quad \sigma_2 = \frac{F_2}{A_2}$$

Cost function: Minimize total mass of the bars, $\text{Mass} = \rho l(A_1 + A_2)$

Constraints:

$$\text{Bar stress: } -\sigma_1 \leq \sigma_1; \quad -\sigma_2 \leq \sigma_2; \quad \sigma_2 \leq \sigma_a$$

$$\text{Design variable limits: } x_{il} \leq x_i \leq x_{iu}; \quad i = 1 \text{ to } 6$$

Note that some of the intermediate variables, such as A_1 , A_2 , F_1 , F_2 , σ_1 , and σ_2 , may also be treated as optimization variables. However, in that case, we will have six equality constraints between the variables, in addition to the other constraints.

EXAMPLE: Formulation with Design Variables Only

Summary of the problem formulation for optimum design of the two-bar bracket in terms of design variables only is as follows:

Specified data: $W, \theta, \sigma_a > 0, x_{il}$ and $x_{iu}, i = 1$ to 6

Design variables: $x_1, x_2, x_3, x_4, x_5, x_6$

Cost function: Minimize total mass of the bars

$$\text{Mass} = \frac{\pi \rho}{4} \sqrt{x_1^2 + (0.5x_2)^2} (x_3^2 - x_4^2 + x_5^2 - x_6^2)$$

Constraints:

Bar stress:

$$\frac{2W \sqrt{x_1^2 + (0.5x_2)^2}}{\pi(x_3^2 - x_4^2)} \left[\frac{\sin \theta}{x_1} + \frac{2 \cos \theta}{x_2} \right] \leq \sigma_a$$

$$\frac{2W \sqrt{x_1^2 + (0.5x_2)^2}}{\pi(x_5^2 - x_6^2)} \left[\frac{\sin \theta}{x_1} - \frac{2 \cos \theta}{x_2} \right] \leq \sigma_a$$

$$\frac{-2W \sqrt{x_1^2 + (0.5x_2)^2}}{\pi(x_5^2 - x_6^2)} \left[\frac{\sin \theta}{x_1} - \frac{2 \cos \theta}{x_2} \right] \leq \sigma_a$$

Design variable limits: $x_{il} \leq x_i \leq x_{iu}; \quad i = 1$ to 6

2.6 Design of a Cabinet

Step 1: Project/Problem Statement A cabinet is assembled from components C_1 , C_2 , and C_3 . Each cabinet requires 8 C_1 , 5 C_2 , and 15 C_3 components. The assembly of C_1 requires either 5 bolts or 5 rivets, whereas C_2 requires 6 bolts or 6 rivets, and C_3 requires 3 bolts or 3 rivets. The cost of installing a bolt, including the cost of the bolt, is \$0.70 for C_1 , \$1.00 for C_2 , and \$0.60 for C_3 . Similarly, riveting costs are \$0.60 for C_1 , \$0.80 for C_2 , and \$1.00 for C_3 . Bolting and riveting capacities per day are 6000 and 8000, respectively. In order to minimize the cost for the 100 cabinets that must be assembled each day, we wish to determine the number of components to be bolted and riveted (after Siddall, 1972).

Step 2: Data and Information Collection All data for the problem are given in the project statement. This problem can be formulated in several different ways depending on the assumptions made. Three formulations are presented, and for each formulation, proper design variables are identified and expressions for the cost and constraint functions are derived, that is, steps 3 to 5 are presented.

2.6.1 Formulation 1 for Cabinet Design

Step 3: Identification/Definition of Design Variables In the first formulation, the following design variables are identified for 100 cabinets:

- x_1 = number of C_1 to be bolted for 100 cabinets
- x_2 = number of C_1 to be riveted for 100 cabinets
- x_3 = number of C_2 to be bolted for 100 cabinets
- x_4 = number of C_2 to be riveted for 100 cabinets
- x_5 = number of C_3 to be bolted for 100 cabinets
- x_6 = number of C_3 to be riveted for 100 cabinets

Step 4: Identification of a Criterion to Be Optimized The design objective is to minimize the total cost of cabinet fabrication, which is obtained from the specified costs for bolting and riveting each component:

$$\begin{aligned} \text{Cost} &= 0.70(5)x_1 + 0.60(5)x_2 + 1.00(6)x_3 + 0.80(6)x_4 + 0.60(3)x_5 + 1.00(3)x_6 \\ &= 3.5x_1 + 3.0x_2 + 6.0x_3 + 4.8x_4 + 1.8x_5 + 3.0x_6 \end{aligned} \quad (\text{a})$$

Step 5: Identification of Constraints The constraints for the problem consist of riveting and bolting capacities and the number of cabinets fabricated each day. Since 100 cabinets must be fabricated, the required numbers of C_1 , C_2 , and C_3 are given in the following constraints:

$$\begin{aligned} x_1 + x_2 &= 8 \times 100 && \text{(number of } C_1 \text{ needed)} \\ x_3 + x_4 &= 5 \times 100 && \text{(number of } C_2 \text{ needed)} \\ x_5 + x_6 &= 15 \times 100 && \text{(number of } C_3 \text{ needed)} \end{aligned} \quad (\text{b})$$

Bolting and riveting capacities must not be exceeded. Thus,

$$\begin{aligned} 5x_1 + 6x_3 + 3x_5 &\leq 6000 && \text{(bolting capacity)} \\ 5x_2 + 6x_4 + 3x_6 &\leq 8000 && \text{(riveting capacity)} \end{aligned} \quad (\text{c})$$

Finally, all design variables must be nonnegative to find a meaningful solution:

$$x_i \geq 0; i = 1 \text{ to } 6 \quad (\text{d})$$

2.6.2 Formulation 2 for Cabinet Design

Step 3: Identification/Definition of Design Variables If we relax the constraint that each component must be bolted or riveted, then the following design variables can be defined:

- x_1 = total number of bolts required for all C_1
- x_2 = total number of bolts required for all C_2
- x_3 = total number of bolts required for all C_3
- x_4 = total number of rivets required for all C_1
- x_5 = total number of rivets required for all C_2
- x_6 = total number of rivets required for all C_3

Step 4: Identification of a Criterion to Be Optimized The objective is still to minimize the total cost of fabricating 100 cabinets, given as

$$\text{Cost} = 0.70x_1 + 1.00x_2 + 0.60x_3 + 0.60x_4 + 0.80x_5 + 1.00x_6 \quad (\text{e})$$

Step 5: Identification of Constraints Since 100 cabinets must be built every day, it will be necessary to have 800 C_1 , 500 C_2 , and 1500 C_3 components. The total number of bolts and rivets needed for all C_1 , C_2 , and C_3 components is indicated by the following equality constraints:

$$\begin{aligned} x_1 + x_4 &= 5 \times 800 && \text{(bolts and rivets needed for } C_1) \\ x_2 + x_5 &= 6 \times 500 && \text{(bolts and rivets needed for } C_2) \\ x_3 + x_6 &= 3 \times 1500 && \text{(bolts and rivets needed for } C_3) \end{aligned} \quad (\text{f})$$

Constraints on capacity for bolting and riveting are:

$$\begin{aligned} x_1 + x_2 + x_3 &\leq 6000 && \text{(bolting capacity)} \\ x_4 + x_5 + x_6 &\leq 8000 && \text{(riveting capacity)} \end{aligned} \quad (\text{g})$$

Finally, all design variables must be nonnegative:

$$x_i \geq 0; i = 1 \text{ to } 6 \quad (\text{h})$$

Thus, this formulation also has six design variables, three equality, and two inequality constraints. After an optimum solution has been obtained, we can decide on how many components to bolt and how many to rivet.

2.6.3 Formulation 3 for Cabinet Design

Step 3: Identification/Definition of Design Variables Another formulation of the problem is possible if we require that all cabinets be identical. The following design variables can be identified:

- x_1 = number of C_1 to be bolted on one cabinet
- x_2 = number of C_1 to be riveted on one cabinet
- x_3 = number of C_2 to be bolted on one cabinet
- x_4 = number of C_2 to be riveted on one cabinet
- x_5 = number of C_3 to be bolted on one cabinet
- x_6 = number of C_3 to be riveted on one cabinet

Step 4: Identification of a Criterion to Be Optimized With these design variables, the cost of fabricating 100 cabinets each day is given as

$$\begin{aligned} \text{Cost} &= 100[0.70(5)x_1 + 0.60(5)x_2 + 1.00(6)x_3 + 0.80(6)x_4 + 0.60(3)x_5 + 1.00(3)x_6] \\ &= 350x_1 + 300x_2 + 600x_3 + 480x_4 + 180x_5 + 300x_6 \end{aligned} \quad (i)$$

Step 5: Identification of Constraints Since each cabinet needs 8 C₁, 5 C₂, and 15 C₃ components, the following equality constraints can be identified:

$$\begin{aligned} x_1 + x_2 &= 8 && \text{(number of C}_1 \text{ needed)} \\ x_3 + x_4 &= 5 && \text{(number of C}_2 \text{ needed)} \\ x_5 + x_6 &= 15 && \text{(number of C}_3 \text{ needed)} \end{aligned} \quad (j)$$

Constraints on the capacity to rivet and bolt are expressed as the following inequalities:

$$\begin{aligned} (5x_1 + 6x_3 + 3x_5)100 &\leq 6000 && \text{(bolting capacity)} \\ (5x_2 + 6x_4 + 3x_6)100 &\leq 8000 && \text{(riveting capacity)} \end{aligned} \quad (k)$$

Finally, all design variables must be nonnegative:

$$x_i \geq 0; i = 1 \text{ to } 6 \quad (l)$$

The following points are noted for these three formulations:

1. Because cost and constraint functions are *linear* in all three formulations, they are linear programming problems. It is conceivable that each formulation will yield a different optimum solution. After solving the problems, the designer can select the best strategy for fabricating cabinets.
2. All formulations have *three equality constraints*, each involving two design variables. Using these constraints, we can eliminate three variables from the problem and thus reduce its dimension. This is desirable from a computational standpoint because the number of variables and constraints is reduced. However, because the elimination of variables is not possible for many complex problems, we must develop methods to treat both equality and inequality constraints.
3. For a meaningful solution with these formulations, all design variables must have integer values. These are called *integer programming problems*. Some numerical methods to treat this class of problem are discussed in Chapter 15.

2.7 Minimum Weight Tubular Column Design

Step 1: Project/Problem Statement Straight columns are used as structural elements in many civil, mechanical, aerospace, agricultural and automotive structures. Many such applications can be observed in daily life, e.g., a street light pole, traffic light post, flag pole, water tower support, highway sign post, power transmission pole, and so on. It is important to optimize the design of a straight column since it may be mass produced. The objective of this project is to design a minimum-mass *tubular* column of length l supporting a load P without buckling or overstressing. The column is fixed at the base and free at the top, as shown in Fig. 2-4. This type of structure is called a cantilever column.

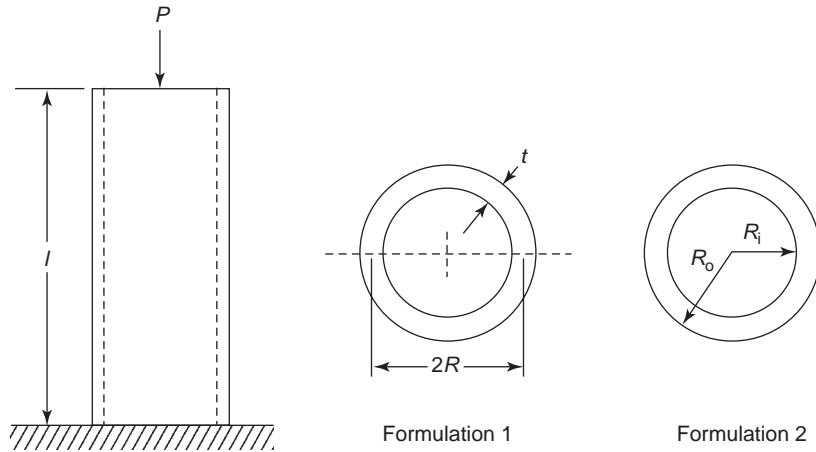


FIGURE 2-4 Tubular column.

Step 2: Data and Information Collection The buckling load (also called the critical load) for a cantilever column is given as

$$P_{cr} = \frac{\pi^2 EI}{4l^2} \quad (a)$$

The buckling load for a column with other support conditions will be different from this formula (Crandall, Dahl, and Lardner, 1978). Here, I is the moment of inertia for the cross section of the column and E is the material property, called the modulus of elasticity (Young's modulus). The material stress σ for the column is defined as P/A , where A is the cross-sectional area of the column. The material allowable stress under the axial load is σ_a , and the material mass density is ρ (mass per unit volume).

A cross section of the tubular column is shown in Fig. 2-4. Many formulations for the design problem are possible depending on how the design variables are defined. Two such formulations are described below.

2.7.1 Formulation 1 for Column Design

Step 3: Identification/Definition of Design Variables For the first formulation, the following design variables are defined:

R = mean radius of the column
 t = wall thickness

Assuming that the column wall is thin ($R \gg t$), the material cross-sectional area and moment of inertia are:

$$A = 2\pi R t; \quad I = \pi R^3 t \quad (b)$$

Step 4: Identification of a Criterion to Be Optimized The total mass of the column to be minimized is given as

$$Mass = \rho(lA) = 2\rho l\pi R t \quad (c)$$

Step 5: Identification of Constraints The first constraint is that the stress (P/A) should not exceed material allowable stress σ_a , to avoid crushing the material. This is expressed as the inequality $\sigma \leq \sigma_a$. Replacing σ by P/A and then substituting for A , we obtain

$$\frac{P}{2\pi R t} \leq \sigma_a \quad (d)$$

The column should not buckle under the applied load P , which implies that the applied load should not exceed the buckling load, i.e., $P \leq P_{cr}$. Using the given expression for the buckling load and substituting for I , we obtain

$$P \leq \frac{\pi^3 E R^3 t}{4l^2} \quad (e)$$

Finally, the design variables R and t must be within the specified minimum and maximum values:

$$R_{\min} \leq R \leq R_{\max}; \quad t_{\min} \leq t \leq t_{\max} \quad (f)$$

2.7.2 Formulation 2 for Column Design

Step 3: Identification/Definition of Design Variables Another formulation of the design problem is possible, if the following design variables are defined:

R_o = outer radius of the column
 R_i = inner radius of the column

In terms of these design variables, the cross-sectional area A and moment of inertia I are:

$$A = \pi(R_o^2 - R_i^2); \quad I = \frac{\pi}{4}(R_o^4 - R_i^4). \quad (g)$$

Step 4: Identification of a Criterion to Be Optimized Minimize the total mass of the column:

$$Mass = \rho(lA) = \pi \rho l (R_o^2 - R_i^2) \quad (h)$$

Step 5: Identification of Constraints The material crushing constraint is ($P/A \leq \sigma_a$)

$$\frac{P}{\pi(R_o^2 - R_i^2)} \leq \sigma_a \quad (i)$$

Using the foregoing expression for I , the buckling load constraint is:

$$P \leq \frac{\pi^3 E}{16l^3} (R_o^4 - R_i^4) \quad (j)$$

Finally, the design variables R_o and R_i must be within specified limits:

$$R_{o \min} \leq R_o \leq R_{o \max}; \quad R_{i \min} \leq R_i \leq R_{i \max} \quad (k)$$

When this problem is solved using a numerical method, a constraint $R_o > R_i$ must also be imposed. Otherwise, some methods may take the design to the point where $R_o < R_i$. This situation is not physically possible and must be explicitly excluded to numerically solve the design problem.

Note that in the second formulation, the assumption of thin-walled sections is not imposed. Thus, optimum solutions with the two formulations can differ. If required, the assumption of thin-walled sections must be explicitly imposed by requiring the ratio between mean radius and wall thickness to be larger than a specified constant k :

$$\frac{(R_o + R_i)}{2(R_o - R_i)} \geq k \quad (1)$$

Usually $k \geq 20$ provides a reasonable approximation for thin-walled sections.

2.8 Minimum Cost Cylindrical Tank Design

Step 1: Project/Problem Statement Design a minimum cost cylindrical tank closed at both ends to contain a fixed volume of fluid V . The cost is found to depend directly on the area of sheet metal used.

Step 2: Data and Information Collection Let c be the dollar cost per unit area of the sheet metal. Other data are given in the project statement.

Step 3: Identification/Definition of Design Variables The design variables for the problem are identified as

R = radius of the tank
 H = height of the tank

Step 4: Identification of a Criterion to Be Optimized The cost function for the problem is the dollar cost of the sheet metal for the tank. Total surface area of the sheet metal consisting of the end plates and cylinder is given as

$$A = 2\pi R^2 + 2\pi RH \quad (a)$$

Therefore, the cost function for the problem is given as

$$f = c(2\pi R^2 + 2\pi RH) \quad (b)$$

Step 5: Identification of Constraints The volume of the tank ($\pi R^2 H$) is required to be V . Therefore,

$$\pi R^2 H = V \quad (c)$$

Also, both design variables R and H must be within some minimum and maximum values:

$$R_{\min} \leq R \leq R_{\max}; \quad H_{\min} \leq H \leq H_{\max} \quad (d)$$

This problem is quite similar to the can problem discussed in Section 2.2. The only difference is in the volume constraint. There the constraint is an inequality and here it is equality.

2.9 Design of Coil Springs

Step 1: Project/Problem Statement Coil springs are used in numerous practical applications. Detailed methods for analyzing and designing such mechanical components have been developed over the years (e.g., Spotts, 1953; Wahl, 1963; Shigley, 1977; Haug and Arora, 1979). The purpose of this project is to design a minimum mass spring (shown in Fig. 2-5) to carry a given axial load (called tension-compression spring) without material failure and while satisfying two performance requirements: the spring must deflect by at least Δ (in.) and the frequency of surge waves must not be less than ω_0 (Hertz, Hz).

Step 2: Data and Information Collection To formulate the problem of designing a coil spring, the following notation is defined:

Deflection along the axis of the spring	δ , in.
Mean coil diameter	D , in.
Wire diameter	d , in.
Number of active coils	N
Gravitational constant	$g = 386 \text{ in./s}^2$
Frequency of surge waves	ω , Hz
Let the material properties be given as	
Weight density of spring material	$\gamma = 0.285 \text{ lb/in.}^3$
Shear modulus	$G = (1.15 \times 10^7) \text{ lb/in.}^2$
Mass density of material ($\rho = \gamma/g$)	$\rho = (7.38342 \times 10^{-4}) \text{ lb-s}^2/\text{in.}^4$
Allowable shear stress	$\tau_a = 80,000 \text{ lb/in.}^2$

Other data for the problem are given as

Number of inactive coils	$Q = 2$
Applied load	$P = 10 \text{ lb}$
Minimum spring deflection	$\Delta = 0.5 \text{ in.}$
Lower limit on surge wave frequency	$\omega_0 = 100 \text{ Hz}$
Limit on outer diameter of the coil	$D_o = 1.5 \text{ in.}$

When the spring is under tension or compression, the wire twists. Therefore, shear stress needs to be calculated so that a constraint on it can be included in the formulation. In addition, surge wave frequency needs to be calculated. The design equations for the spring are given as

$$\text{Load deflection equation: } P = K\delta \quad (\text{a})$$

$$\text{Spring constant: } K = \frac{d^4 G}{8D^3 N} \quad (\text{b})$$

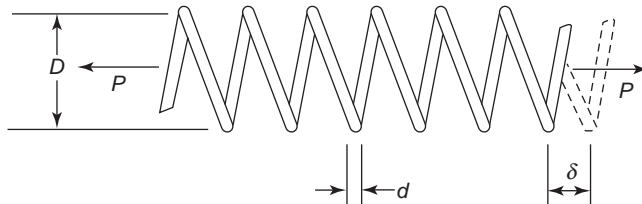


FIGURE 2-5 A coil spring.

$$\text{Shear stress:} \quad \tau = \frac{8kPD}{\pi d^3} \quad (c)$$

$$\text{Wahl stress concentration factor:} \quad k = \frac{(4D-d)}{4(D-d)} + \frac{0.615d}{D} \quad (d)$$

$$\text{Frequency of surge waves:} \quad \omega = \frac{d}{2\pi ND^2} \sqrt{\frac{G}{2\rho}} \quad (e)$$

The expression for the Wahl stress concentration factor k in Eq. (d) has been determined experimentally to account for unusually high stresses at certain points on the spring. These analysis equations are used to define the constraints.

Step 3: Identification/Definition of Design Variables The three design variables for the problem are defined as

- d = wire diameter, in.
- D = mean coil diameter, in.
- N = number of active coils

Step 4: Identification of a Criterion to Be Optimized The problem is to *minimize the mass* of the spring, given as volume \times mass density:

$$\text{Mass} = \frac{1}{4}(N+Q)\pi^2 Dd^2 \rho \quad (f)$$

Step 5: Identification of Constraints *Deflection constraint.* It is often a requirement that *deflection* under a load P be at least Δ . Therefore, the constraint is that the calculated deflection δ must be greater or equal to Δ . Such a constraint is common to spring design. The function of the spring in many applications is to provide a modest restoring force as parts undergo large displacement in carrying out kinematic functions. Mathematically, this performance requirement ($\delta \geq \Delta$) is stated in an inequality form using Eq. (a), and as

$$\frac{P}{K} \geq \Delta \quad (g)$$

Shear stress constraint. To prevent material overstressing, *shear stress* in the wire must be no greater than τ_a , which is expressed in mathematical form as

$$\tau \leq \tau_a \quad (h)$$

Constraint on frequency of surge waves. We also wish to avoid resonance in dynamic applications by making the *frequency of surge waves* (along the spring) as large as possible. For the present problem, we require the frequency of surge waves for the spring to be at least ω_0 (Hz). The constraint is expressed in mathematical form as

$$\omega \geq \omega_0 \quad (i)$$

Diameter constraint. The *outer diameter* of the spring should not be greater than D_{os} , so

$$D + d \leq D_0 \quad (j)$$

Explicit bounds on design variables. To avoid fabrication and other practical difficulties, we put *minimum and maximum size limits* on the wire diameter, coil diameter, and number of turns:

$$\begin{aligned} d_{\min} &\leq d \leq d_{\max} \\ D_{\min} &\leq D \leq D_{\max} \\ N_{\min} &\leq N \leq N_{\max} \end{aligned} \quad (k)$$

Thus, the purpose of the minimum mass spring design problem is to select the design variables d , D , and N to minimize the mass of Eq. (f), while satisfying the ten inequality constraints of Eqs. (g) through (k). If the intermediate variables are eliminated, the problem formulation can be summarized in terms of the design variables only.

EXAMPLE: Formulation with Design Variables Only

Summary of the problem formulation for optimum design of coil springs is as follows:

Specified data: $Q, P, \rho, \gamma, \tau_a, G, \Delta, \omega_0, D_0, d_{\min}, d_{\max}, D_{\min}, D_{\max}, N_{\min}, N_{\max}$

Design variables: d, D, N

Cost function: Minimize

$$Mass = \frac{1}{4}(N + Q)\pi^2 D d^2 \rho$$

Constraints:

$$\text{Deflection limit:} \quad \frac{8PD^3N}{d^4G} \geq \Delta$$

$$\text{Shear stress:} \quad \frac{8PD}{\pi d^3} \left[\frac{(4D-d)}{4(D-d)} + \frac{0.615d}{D} \right] \leq \tau_a$$

$$\text{Frequency of surge waves:} \quad \frac{d}{2\pi ND^2} \sqrt{\frac{G}{2\rho}} \geq \omega_0$$

$$\text{Diameter constraint:} \quad D + d \leq D_0$$

$$\text{Design variable bounds:} \quad d_{\min} \leq d \leq d_{\max}$$

$$D_{\min} \leq D \leq D_{\max}$$

$$N_{\min} \leq N \leq N_{\max}$$

2.10 Minimum Weight Design of a Symmetric Three-Bar Truss

Step 1: Project/Problem Statement As an example of a slightly more complex design problem, consider the three-bar structure shown in Fig. 2-6 (Schmit 1960; Haug and Arora 1979). The structure is to be designed for minimum volume (or, equivalently, minimum mass)

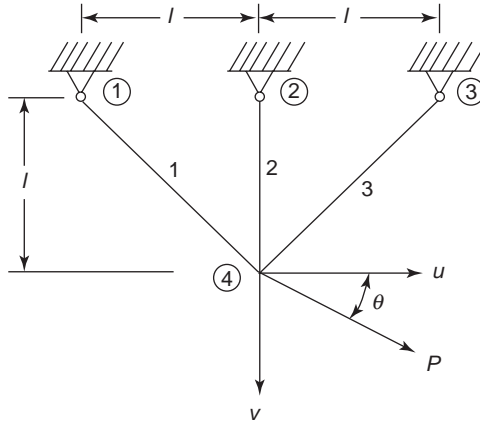


FIGURE 2-6 Three-bar truss.

to support a force P . It must satisfy various performance and technological constraints, such as member crushing, member buckling, failure by excessive deflection of node 4, and failure by resonance when natural frequency of the structure is below a given threshold.

Step 2: Data and Information Collection The data needed to solve the problem are: geometry data, properties of the material used, and the loading data. In addition, since the structure is statically indeterminate, we need to use advanced analysis procedures to obtain expressions for member forces, nodal displacements, and the natural frequency to formulate constraints for the problem. Here we shall give such expressions.

Since the structure must be symmetric, members 1 and 3 will have same cross-sectional area, say A_1 . Let A_2 be the cross-sectional area of member 2. Using analysis procedures for statically indeterminate structures, horizontal and vertical displacements u and v of node 4 are:

$$u = \frac{\sqrt{2}lP_u}{A_1E}; \quad v = \frac{\sqrt{2}lP_v}{(A_1 + \sqrt{2}A_2)E} \quad (a)$$

where E is the modulus of elasticity for the material, P_u and P_v are the horizontal and vertical components of the load P given as $P_u = P \cos \theta$ and $P_v = P \sin \theta$, and l is defined in Fig. 2-6. Using the displacements, forces carried by the members of the truss can be calculated. Then the stresses σ_1 , σ_2 , and σ_3 in members 1, 2, and 3 under the applied load P in Fig. 2-6 can be computed from member forces as (stress = force/area)

$$\sigma_1 = \frac{1}{\sqrt{2}} \left[\frac{P_u}{A_1} + \frac{P_v}{(A_1 + \sqrt{2}A_2)} \right] \quad (b)$$

$$\sigma_2 = \frac{\sqrt{2}P_v}{(A_1 + \sqrt{2}A_2)} \quad (c)$$

$$\sigma_3 = \frac{1}{\sqrt{2}} \left[-\frac{P_u}{A_1} + \frac{P_v}{(A_1 + \sqrt{2}A_2)} \right] \quad (d)$$

Many structures support moving machinery and other dynamic loads. These structures vibrate with a certain frequency known as the *natural frequency*. This is an intrinsic dynamic

property of a structural system. There can be several modes of vibration each having its own frequency. *Resonance* causes catastrophic failure of the structure, which occurs when any of its vibration frequency coincides with the frequency of the operating machinery it supports. Therefore, it is reasonable to demand that no structural frequency be close to the frequency of the operating machinery. The mode of vibration corresponding to the lowest natural frequency is important because that mode is excited first. It is important to make the lowest (fundamental) natural frequency of the structure as high as possible to avoid any possibility of resonance. This also makes the structure stiffer. Frequencies of a structure are obtained by solving an eigenvalue problem involving its stiffness and mass properties. The lowest eigenvalue ζ related to the lowest natural frequency of the symmetric three-bar truss is computed using a consistent mass model:

$$\zeta = \frac{3EA_1}{\rho l^2(4A_1 + \sqrt{2}A_2)} \quad (e)$$

where ρ is the material mass per unit volume (mass density). This completes analysis of the structure.

Step 3: Identification/Definition of Design Variables The following design variables are defined for the symmetric structure:

A_1 = cross-sectional area of material for members 1 and 3

A_2 = cross-sectional area of material for member 2

Other design variables for the problem are possible depending on the cross-sectional shape of members, as shown in Fig. 2-3.

Step 4: Identification of a Criterion to Be Optimized The relative merit of any design for the problem is measured in its material volume. Therefore, total volume of the structural material serves as a cost function (volume of a member = cross-sectional area \times length):

$$\text{Volume} = l(2\sqrt{2}A_1 + A_2) \quad (f)$$

Step 5: Identification of Constraints The structure is designed for use in two applications. In each application, it supports different loads. These are called loading conditions for the structure. In the present application, a symmetric structure would be obtained if the following two loading conditions are considered for the structure. The first load is applied at an angle θ and the second one at an angle $(\pi - \theta)$, where the angle θ ($0^\circ \leq \theta \leq 90^\circ$) is shown in Fig. 2-6. If we let member 1 be the same as member 3, then the second loading condition can be ignored. Therefore, we consider only one load applied at an angle θ ($0^\circ \leq \theta \leq 90^\circ$).

Note from Eqs. (b) and (d) that σ_1 is always larger than σ_3 . Therefore, we need to impose constraints on only σ_1 and σ_2 . If σ_a is an allowable stress for the material, then the *stress constraints* are:

$$\sigma_1 \leq \sigma_a; \quad \sigma_2 \leq \sigma_a \quad (g)$$

Horizontal and vertical deflections of node 4 must be within the specified limits Δ_u and Δ_v , respectively. Using Eq. (a), the *deflection constraints* are:

$$u \leq \Delta_u; \quad v \leq \Delta_v \quad (h)$$

As discussed previously, the *fundamental natural frequency* of the structure should be higher than a specified frequency ω_0 (Hz). This constraint can be written in terms of the lowest eigenvalue for the structure. The eigenvalue corresponding to a frequency of ω_0 (Hz) is given as $(2\pi\omega_0)^2$. The lowest eigenvalue ζ for the structure should be higher than $(2\pi\omega_0)^2$, i.e.,

$$\zeta \geq (2\pi\omega_0)^2 \quad (i)$$

To impose *buckling constraints* for members under compression, an expression for the moment of inertia of the cross section is needed. This expression cannot be obtained since the cross-sectional shape and dimensions are not specified. However, the moment of inertia I can be related to the cross-sectional area of the members as $I = \beta A^2$, where A is the cross-sectional area and β is a nondimensional constant. This relation follows if the shape of the cross section is fixed and all its dimensions are varied in the same proportion. The axial force for the i th member is given as $F_i = A_i \sigma_i$, where $i = 1, 2, 3$ with tensile force taken as positive. Members of the truss are considered columns with pin ends. Therefore, the buckling load for the i th member is given as $\pi^2 EI/l_i^2$, where l_i is the length of the i th member (Crandall, Dahl, and Lardner, 1978). Buckling constraints are expressed as $-F_i \leq \pi^2 EI/l_i^2$, where $i = 1, 2, 3$. The negative sign for F_i is used to make the left side of the constraints positive when the member is in compression. Also, there is no need to impose buckling constraints for members in tension. With the foregoing formulation, the buckling constraint for tensile members is automatically satisfied. Substituting various quantities, member buckling constraints are:

$$-\sigma_1 \leq \frac{\pi^2 E \beta A_1}{2l^2}; \quad -\sigma_2 \leq \frac{\pi^2 E \beta A_2}{l^2}; \quad -\sigma_3 \leq \frac{\pi^2 E \beta A_1}{2l^2} \quad (j)$$

Note that the buckling load on the right side has been divided by the member area in the foregoing expressions. Also, the first two constraints in Eq. (j) are automatically satisfied since both the members are always in tension (forces in them are always positive for the direction of load shown in Fig. 2-6).

Finally, A_1 and A_2 must both be nonnegative, i.e., $A_1, A_2 \geq 0$. Most practical design problems would require each member to have a certain minimum area, A_{\min} . The minimum area constraints can be written as

$$A_1, A_2 \geq A_{\min} \quad (k)$$

The optimum design problem then is to find cross-sectional areas $A_1, A_2 \geq A_{\min}$ to minimize the volume of Eq. (f) subject to the constraints of Eqs. (g) to (k). This small-scale problem has 10 inequality constraints and 2 design variables.

2.11 A General Mathematical Model for Optimum Design

To describe optimization concepts and methods, we need a general mathematical statement for the optimum design problem. Such a mathematical model is defined as minimization of a cost function while satisfying all the equality and inequality constraints. The inequality constraints in the model are always transformed as “ \leq types.” This will be called the *standard design optimization model* that is treated throughout this text. It will be shown that all design problems can easily be transcribed into the standard form.

2.11.1 Standard Design Optimization Model

In previous sections, several design problems were formulated. All problems have an optimization criterion that can be used to compare various designs and determine an optimum (the best) one. Most design problems must also satisfy certain constraints. Some design problems have only inequality constraints, others have only equality constraints, and some have both inequalities and equalities. We can define a general mathematical model for optimum design to encompass all the possibilities. A standard form of the model is first stated and then transformation of various problems into the standard form is explained.

Standard Design Optimization Model Find an n -vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$ of design variables to *minimize a cost function*

$$f(\mathbf{x}) = f(x_1, x_2, \dots, x_n) \quad (2.1)$$

subject to the p equality constraints

$$h_j(\mathbf{x}) = h_j(x_1, x_2, \dots, x_n) = 0; \quad j = 1 \text{ to } p \quad (2.2)$$

and the m inequality constraints

$$g_i(\mathbf{x}) = g_i(x_1, x_2, \dots, x_n) \leq 0; \quad i = 1 \text{ to } m \quad (2.3)$$

Note that the simple bounds on design variables, such as $x_i \geq 0$, or $x_{il} \leq x_i \leq x_{iu}$, where x_{il} and x_{iu} are the smallest and largest allowed value for x_i , are assumed to be included in the inequalities of Eq. (2.3). In numerical methods, these constraints are treated explicitly to take advantage of their simple form to effect efficiency. However, in discussing the basic optimization concepts, we assume that the inequalities in Eq. (2.3) include these constraints as well.

Application to Different Engineering Fields Design optimization problems from different fields of engineering can be transcribed into the standard model. *It must be realized that the overall process of designing different engineering systems is the same.* Analytical and numerical methods for analyzing systems can differ. Formulation of the design problem can contain terminology that is specific to the particular domain of application. For example, in the fields of structural, mechanical, and aerospace engineering, we are concerned with the integrity of the structure and its components. The performance requirements involve constraints on member stresses, strains, deflections at key points, frequencies of vibration, buckling failure, and so on. These terms are specific to the fields, and designers working in the area understand their meaning and the constraints. Similarly, other fields of engineering have their own terminology to describe design optimization problems. However, *once the problems from different fields have been transcribed into mathematical statements using a standard notation, they have the same mathematical form.* They are contained in the standard design optimization model defined in Eqs. (2.1) to (2.3). For example, all the problems formulated earlier in this chapter can be transformed into the form of Eqs. (2.1) to (2.3). Therefore, *optimization methods described in the text are quite general and can be used to solve problems from diverse fields. The methods can be developed without reference to any design application.* This *key point* must be kept in mind while studying the optimization concepts and the methods.

Important Observations about Standard Model Several points must clearly be understood about the standard model:

1. First of all, it is obvious that the functions $f(\mathbf{x})$, $h_j(\mathbf{x})$, and $g_i(\mathbf{x})$ must *depend*, explicitly or implicitly, on some of the *design variables*. Only then are they valid for the design problem. Functions that do not depend on any variable have no relation to the problem and can be safely ignored.
2. The number of *independent equality constraints* must be less than or at the most equal to the number of design variables, i.e., $p \leq n$. When $p > n$, we have an *over-determined system* of equations. In that case, either there are some *redundant equality constraints* (linearly dependent on other constraints), or they are *inconsistent*. In the former case, redundant constraints can be deleted and, if $p < n$, the optimum solution for the problem is possible. In the latter case, no solution for the design problem is possible and the problem formulation should be closely re-examined. When $p = n$, no optimization of the system is necessary because solutions of the equality constraints are the only candidates for optimum design.
3. Note that all inequality constraints in Eq. (2.3) are written as “ ≤ 0 .” This is standard practice throughout the text. In the example problems of previous sections, we encountered “ \leq type” as well as “ \geq type” constraints. “ \leq type” constraints can be converted to the standard form of Eq. (2.3) by transferring the right side to the left side. “ \geq type” constraints can also be transformed to the “ \leq form” quite easily by multiplying them by -1 as explained later. Note, however, that while there is a restriction on the number of independent equality constraints, *there is no restriction on the number of inequality constraints*. However, the total number of active constraints (satisfied at equality) at the optimum is usually less than or at the most equal to the number of design variables.
4. Some design problems may not have any constraints. These are called *unconstrained* optimization problems, and others are called *constrained* optimization problems.
5. If all the functions $f(\mathbf{x})$, $h_j(\mathbf{x})$, and $g_i(\mathbf{x})$ are linear in design variables \mathbf{x} , then the problem is called a *linear programming problem*. If any of these functions is nonlinear, the problem is called a *nonlinear programming problem*.
6. It is important to note that if the *cost function is scaled* by multiplying it with a positive constant, the optimum design does not change. The optimum cost function value, however, changes. Also, any constant can be added to the cost function without affecting the optimum design. Similarly, the *inequality constraints can be scaled* by any positive constant and equalities by any constant. This will not affect the feasible region and hence the optimum solution. All the foregoing transformations, however, affect the values of the Lagrange multipliers (defined in Chapter 4). Also, performance of the numerical algorithms is affected by these transformations.

2.11.2 Maximization Problem Treatment

The general design model treats only minimization problems. This is no restriction as maximization of a function $F(\mathbf{x})$ is the same as minimization of a transformed function $f(\mathbf{x}) = -F(\mathbf{x})$. To see this graphically, consider a plot of the function of one variable $F(x)$, shown in Fig. 2-7(A). The function $F(x)$ takes its maximum value at the point x^* . Next consider a graph of the function $f(x) = -F(x)$ shown in Fig. 2-7(B). It is clear that $f(x)$ is a reflection of $F(x)$ about the x axis. It is also clear from the graph that $f(x)$ takes on a minimum value at the same point x^* where the maximum of $F(x)$ occurs. Therefore, minimization of $f(x)$ is equivalent to maximization of $F(x)$.

2.11.3 Treatment of “Greater Than Type” Constraints

The standard design optimization model treats only “ \leq type” inequality constraints. Many design problems may also have “ \geq type” inequalities. Such constraints can be converted to

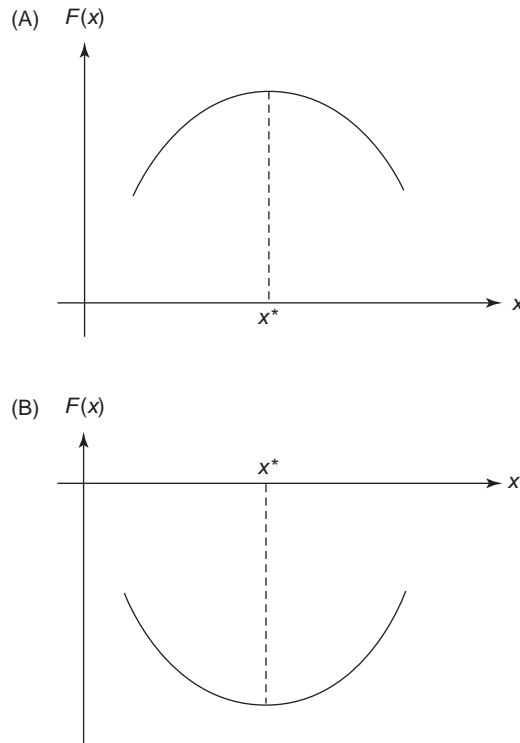


FIGURE 2-7 Point maximizing $F(x)$ equals the point minimizing $-F(x)$. (A) Plot of $F(x)$. (B) Plot of $f(x) = -F(x)$.

the standard form without much difficulty. A “ \geq type” constraint $G_j(\mathbf{x}) \geq 0$ is equivalent to the “ \leq type” inequality $g_j(\mathbf{x}) = -G_j(\mathbf{x}) \leq 0$. Therefore, we can multiply any “ \geq type” constraint by -1 to convert it to the “ \leq type.”

2.11.4 Discrete and Integer Design Variables

So far, we have assumed in the standard model that variables x_i can have any numerical value within the feasible region. Many times, however, some variables are required to have discrete or integer values. Such variables appear quite often in engineering design problems. We have already encountered problems in Sections 2.4 and 2.6 that have integer design variables. Before describing how to treat them, let us define what we mean by discrete and integer variables.

A design variable is called *discrete* if its value must be selected from a given finite set of values. For example, the plate thickness must be the one that is available commercially, i.e., $1/8, 1/4, 3/8, 1/2, 5/8, 3/4, 1, \dots$ and so on. Similarly, structural members must be selected off-the-shelf to reduce the fabrication cost. Such variables must be treated as discrete in the standard formulation. An *integer variable*, as the name implies, must have an integer value, e.g., the number of logs to be shipped, number of bolts used, number of items to be shipped, and so on. These are called *discrete and integer programming problems*. Depending on the type of problem functions, the problems can be classified into five different types. These classifications and methods to solve them are discussed in Chapter 15.

In some sense, discrete and integer variables impose additional constraints on the design problem. Therefore, as noted before, the optimum value of the cost function is likely to

increase with their presence compared with the same problem that is solved with continuous variables. If we treat all the design variables as continuous, the minimum value of the cost function represents a lower bound on the true minimum value when discrete or integer variables are used. This gives some idea of the “best” optimum solution if all design variables were continuous. The optimum cost function value is likely to increase when discrete values are assigned to variables. Thus, the first suggested procedure is to solve the problem assuming continuous design variables if that is possible. Then the nearest discrete/integer values are assigned to the variables and the design is checked for feasibility. With a few trials, the best feasible design close to the continuous optimum can be obtained. Note that there can be numerous combinations of discrete variables that can give feasible designs.

As a second approach, an *adaptive numerical optimization procedure* may be used. An optimum solution with continuous variables is first obtained if that is possible. Then, only the variables that are close to their discrete or integer value are assigned that value. They are then held fixed and the problem is optimized again. The procedure is continued until all the variables have been assigned discrete or integer values. A few further trials may be made to improve the optimum cost function value. This procedure has been demonstrated by Arora and Tseng (1988).

The foregoing procedures require additional computational effort and do not guarantee true minimum solution. However, they are quite straightforward and do not require any additional methods or software for solution of discrete/integer variable problems.

2.11.5 Feasible Set

The term “feasible set” will be used throughout the text. A *feasible set for the design problem is a collection of all feasible designs*. The terms “constraint set” and “feasible design space” are also used to represent the feasible set of designs. The letter S will be used to represent the feasible set. Mathematically, the set S is a collection of design points satisfying all the constraints:

$$S = \{\mathbf{x} | h_j(\mathbf{x}) = 0, j = 1 \text{ to } p; g_i(\mathbf{x}) \leq 0, i = 1 \text{ to } m\} \quad (2.4)$$

The *set of feasible designs* is sometimes referred to as the *feasible region*, especially for optimization problems with two design variables. It is important to note that the *feasible region usually shrinks when more constraints are added in the design model and expands when some constraints are deleted*. When the feasible region shrinks, the number of possible designs that can optimize the cost function is reduced, i.e., there are fewer feasible designs. In this event, the minimum value of the cost function is likely to increase. The effect is completely the opposite when some constraints are dropped. This observation is significant in practical design and should be clearly understood.

2.11.6 Active/Inactive/Violated Constraints

We will quite frequently refer to a constraint as *active*, *tight*, *inactive*, or *violated*. We define these terms precisely. An inequality constraint $g_j(\mathbf{x}) \leq 0$ is said to be *active* at a design point \mathbf{x}^* if it is satisfied at equality, i.e., $g_j(\mathbf{x}^*) = 0$. This will be also called a *tight* or *binding* constraint. For a feasible design, an inequality constraint may or may not be active. However, all equality constraints are active for all feasible designs.

An inequality constraint $g_j(\mathbf{x}) \leq 0$ is said to be *inactive* at a design point \mathbf{x}^* if it is strictly satisfied, i.e., $g_j(\mathbf{x}^*) < 0$. It is said to be *violated* at a design point \mathbf{x}^* if its value is positive, i.e., $g_j(\mathbf{x}^*) > 0$. An *equality constraint* $h_i(\mathbf{x}) = 0$ is violated at a design point \mathbf{x}^* if $h_i(\mathbf{x}^*)$ is not identically zero. Note that by these definitions, an equality constraint is either active or violated at a given design point.

Exercises for Chapter 2

- 2.1 A 100×100 m lot is available to construct a multistory office building. At least $20,000 \text{ m}^2$ total floor space is needed. According to a zoning ordinance, the maximum height of the building can be only 21 m, and the area for parking outside the building must be at least 25 percent of the total floor area. It has been decided to fix the height of each story at 3.5 m. The cost of the building in millions of dollars is estimated at $0.6h + 0.001A$, where A is the cross-sectional area of the building per floor and h is the height of the building. Formulate the minimum cost design problem.
- 2.2 A refinery has two crude oils:
1. Crude A costs \$30/barrel (bbl) and 20,000 bbl are available
 2. Crude B costs \$36/bbl and 30,000 bbl are available.

The company manufactures gasoline and lube oil from the crudes. Yield and sale price per barrel of the product and markets are shown in Table E2-2. How much crude oils should the company use to maximize its profit? Formulate the optimum design problem.

TABLE E2-2 Data for Refinery Operations

Product	Yield/bbl		Sale price per bbl (\$)	Market (bbl)
	Crude A	Crude B		
Gasoline	0.6	0.8	50	20,000
Lube oil	0.4	0.2	120	10,000

- 2.3 Design a beer mug, shown in Fig. E2-3, to hold as much beer as possible. The height and radius of the mug should be not more than 20 cm. The mug must be at least 5 cm in radius. The surface area of the sides must not be greater than 900 cm^2 (ignore the area of the bottom of the mug and ignore the mug handle—see figure). Formulate the optimum design problem.

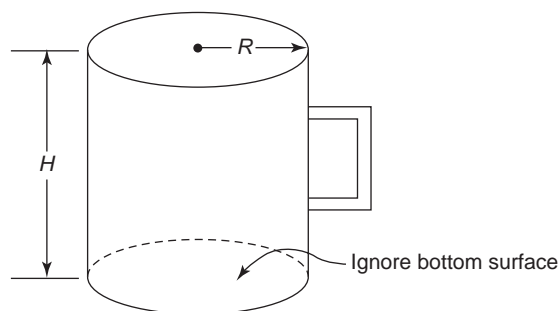


FIGURE E2-3 Beer mug.

- 2.4 A company is redesigning its parallel flow heat exchanger of length l to increase its heat transfer. An end view of the unit is shown in Fig. E2-4. There are certain limitations on the design problem. The smallest available conducting tube has a radius of 0.5 cm and all tubes must be of the same size. Further, the total cross-

sectional area of all the tubes cannot exceed 2000cm^2 to ensure adequate space inside the outer shell. Formulate the problem to determine the number of tubes and the radius of each tube to maximize the surface area of the tubes in the exchanger.

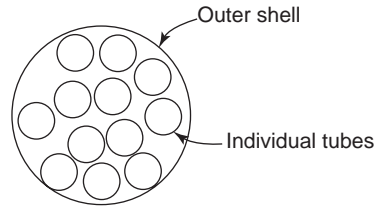


FIGURE E2-4 Cross section of heat exchanger.

- 2.5 Proposals for a parking ramp having been defeated, we plan to build a parking lot in the downtown urban renewal section. The cost of land is $200W + 100D$, where W is the width along the street and D the depth of the lot in meters. The available width along the street is 100 m, while the maximum depth available is 200 m. We want to have at least $10,000\text{m}^2$ in the lot. To avoid unsightly lots, the city requires that the longer dimension of any lot be no more than twice the shorter dimension. Formulate the minimum cost design problem.
- 2.6 A manufacturer sells products A and B. Profit from A is \$10/kg and from B \$8/kg. Available raw materials for the products are: 100 kg of C and 80 kg of D. To produce 1 kg of A, 0.4 kg of C and 0.6 kg of D are needed. To produce 1 kg of B, 0.5 kg of C and 0.5 kg of D are needed. The markets for the products are 70 kg for A and 110 kg for B. How much of A and B should be produced to maximize profit? Formulate the design optimization problem.
- 2.7 Design a diet of bread and milk to get at least 5 units of vitamin A and 4 units of vitamin B each day. The amount of vitamins A and B in 1 kg of each food and the cost per kilogram of food are given in Table E2-7. Formulate the design optimization problem so that we get at least the basic requirements of vitamins at the minimum cost.

TABLE E2-7 Data for the Diet Problem

<i>Vitamin</i>	<i>Bread</i>	<i>Milk</i>
A	1	2
B	3	2
Cost/kg	2	1

- 2.8 Enterprising chemical engineering students have set up a still in a bathtub. They can produce 225 bottles of pure alcohol each week. They bottle two products from alcohol: (i) wine, 20 proof, and (ii) whiskey, 80 proof. Recall that pure alcohol is 200 proof. They have an unlimited supply of water but can only obtain 800 empty bottles per week because of stiff competition. The weekly supply of sugar is enough for either 600 bottles of wine or 1200 bottles of whiskey. They make \$1.00 profit on each bottle of wine and \$2.00 profit on each bottle of whiskey. They can sell whatever they produce. How many bottles of wine and whiskey should they produce each week to maximize profit. Formulate the design optimization problem (created by D. Levy).

- 2.9 Design a can closed at one end using the smallest area of sheet metal for a specified interior volume of 600 m^3 . The can is a right circular cylinder with interior height h and radius r . The ratio of height to diameter must not be less than 1.0 and not greater than 1.5. The height cannot be more than 20 cm. Formulate the design optimization problem.
- 2.10 Design a shipping container closed at both ends with dimensions $b \times b \times h$ to minimize the ratio: (round-trip cost of shipping the container only)/(one-way cost of shipping the contents only). Use the following data:
 Mass of the container/surface area: 80 kg/m^2
 Maximum b : 10 m
 Maximum h : 18 m
 One-way shipping cost,
 full or empty: $\$18/\text{kg}$ gross mass
 Mass of the contents: 150 kg/m^3
 Formulate the design optimization problem.
- 2.11 Certain mining operations require an open top rectangular container to transport materials. The data for the problem are:
 Construction costs:
 sides: $\$50/\text{m}^2$
 ends: $\$60/\text{m}^2$
 bottom: $\$90/\text{m}^2$
 Salvage value: 25 percent of the construction cost
 Useful life: 20 years
 Yearly maintenance: $\$12/\text{m}^2$ of outside surface area
 Minimum volume needed: 150 m^3
 Interest rate: 12 percent per annum
 Formulate the problem of determining the container dimensions for minimum present cost.
- 2.12 Design a circular tank closed at both ends to have a volume of 250 m^3 . The fabrication cost is proportional to the surface area of the sheet metal and is $\$400/\text{m}^2$. The tank is to be housed in a shed with a sloping roof. Therefore, height H of the tank is limited by the relation $H \leq 10 - D/2$, where D is the diameter of the tank. Formulate the minimum cost design problem.
- 2.13 Design the steel framework shown in Fig. E2-13 at a minimum cost. The cost of a horizontal member in one direction is $\$20w$ and in the other direction it is $\$30d$. The cost of a vertical column is $\$50h$. The frame must enclose a total volume of at least 600 m^3 . Formulate the design optimization problem.

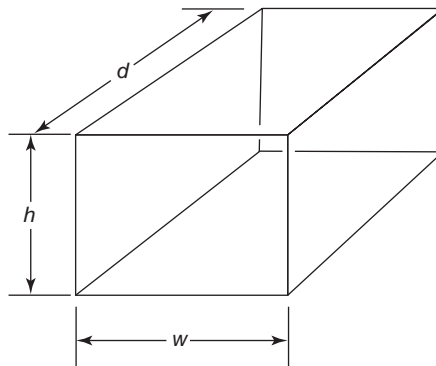


FIGURE E2-13 Steel frame.

- 2.14 Two electric generators are interconnected to provide total power to meet the load. Each generator's cost is a function of the power output, as shown in Fig. E2-14. All costs and power are expressed on a per unit basis. The total power needed is at least 60 units. Formulate a minimum cost design problem to determine the power outputs P_1 and P_2 .

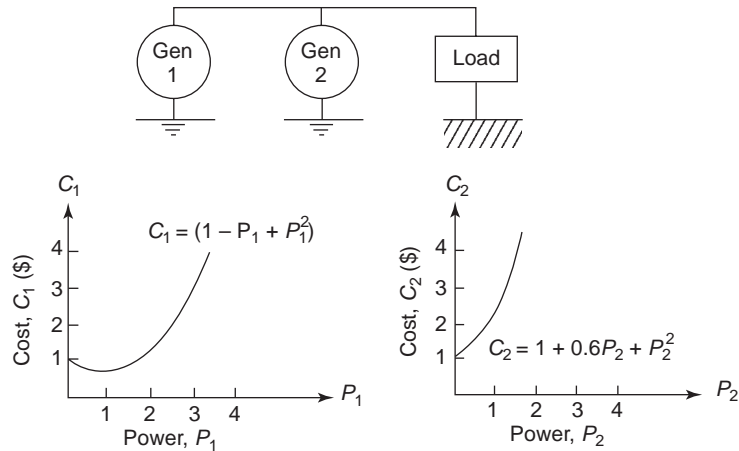


FIGURE E2-14 Power generator.

- 2.15 **Transportation problem.** A company has m manufacturing facilities. The facility at the i th location has capacity to produce b_i units of an item. The product should be shipped to n distribution centers. The distribution center at the j th location requires at least a_j units of the item to satisfy demand. The cost of shipping an item from the i th plant to the j th distribution center is c_{ij} . Formulate a minimum cost transportation system to meet each distribution center's demand without exceeding the capacity of any manufacturing facility.
- 2.16 **Design of a two-bar truss.** Design a symmetric two-bar truss (both members have the same cross section) shown in Fig. E2-16 to support a load W . The truss consists of two steel tubes pinned together at one end and supported on the ground at the other. The span of the truss is fixed at s . Formulate the minimum mass truss design problem using height and the cross-sectional dimensions as design variables. The design should satisfy the following constraints:

1. Because of space limitations, the height of the truss must not exceed b_1 , and must not be less than b_2 .
2. The ratio of the mean diameter to thickness of the tube must not exceed b_3 .
3. The compressive stress in the tubes must not exceed the allowable stress σ_a for steel.
4. The height, diameter, and thickness must be chosen to safeguard against member buckling.

Use the following data: $W = 10$ kN; span $s = 2$ m; $b_1 = 5$ m; $b_2 = 2$ m; $b_3 = 90$; allowable stress, $\sigma_a = 250$ MPa; modulus of elasticity, $E = 210$ GPa; mass density, $\rho = 7850$ kg/m³; factor of safety against buckling, FS = 2; $0.1 \leq D \leq 2$ (m); and $0.01 \leq t \leq 0.1$ (m).

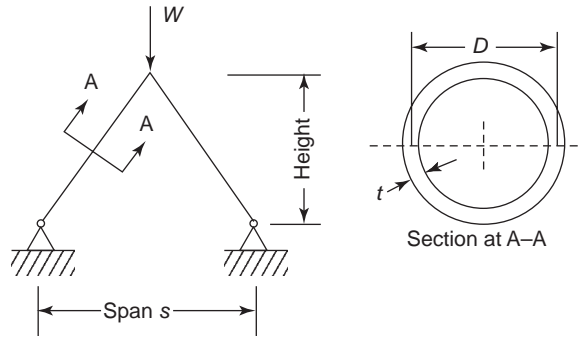


FIGURE E2-16 Two-bar structure.

2.17 A beam of rectangular cross section (Fig. E2-17) is subjected to a maximum bending moment of M and a maximum shear of V . The allowable bending and shearing stresses are σ_a and τ_a , respectively. The bending stress in the beam is calculated as

$$\sigma = \frac{6M}{bd^2}$$

and average shear stress in the beam is calculated as

$$\tau = \frac{3V}{2bd}$$

where d is the depth and b is the width of the beam. It is also desired that the depth of the beam shall not exceed twice its width. Formulate the design problem for minimum cross-sectional area using the following data: $M = 140 \text{ kN} \cdot \text{m}$, $V = 24 \text{ kN}$, $\sigma_a = 165 \text{ MPa}$, $\tau_a = 50 \text{ MPa}$.

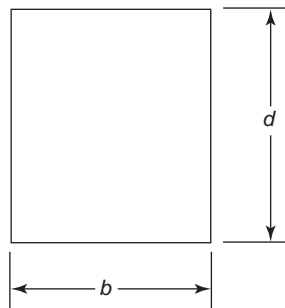


FIGURE E2-17 Cross section of a rectangular beam.

2.18 A vegetable oil processor wishes to determine how much shortening, salad oil, and margarine to produce to optimize the use of his current oil stocks. At the current time, he has 250,000 kg of soybean oil, 110,000 kg of cottonseed oil, and 2000 kg of milk base substances. The milk base substances are required only in the production of margarine. There are certain processing losses associated with each product; 10 percent for shortening, 5 percent for salad oil, and no loss for margarine. The

producer's back orders require him to produce at least 100,000 kg of shortening, 50,000 kg of salad oil, and 10,000 kg of margarine. In addition, sales forecasts indicate a strong demand for all products in the near future. The profit per kilogram and the base stock required per kilogram of each product are given in Table E2-18. Formulate the problem to maximize profit over the next production scheduling period (created by J. Liittschwager).

TABLE E2-18 Data for the Vegetable Oil Processing Problem

Product	Profit per kg	Parts per kg of base stock requirements		
		Soybean	Cottonseed	Milk base
Shortening	0.10	2	1	0
Salad oil	0.08	0	1	0
Margarine	0.05	3	1	1

Section 2.11 A General Mathematical Model for Optimum Design

2.19 *Answer True or False.*

1. Design of a system implies specification for the design variable values.
2. All design problems have only linear inequality constraints.
3. All design variables should be independent of each other as far as possible.
4. If there is an equality constraint in the design problem, the optimum solution must satisfy it.
5. Each optimization problem must have certain parameters called the design variables.
6. A feasible design may violate equality constraints.
7. A feasible design may violate "≥ type" constraints.
8. A "≤ type" constraint expressed in the standard form is active at a design point if it has zero value there.
9. The constraint set for a design problem consists of all the feasible points.
10. The number of independent equality constraints can be larger than the number of design variables for the problem.
11. The number of "≤ type" constraints must be less than the number of design variables for a valid problem formulation.
12. The feasible region for an equality constraint is a subset of that for the same constraint expressed as an inequality.
13. Maximization of $f(x)$ is equivalent to minimization of $1/f(x)$.
14. A lower minimum value for the cost function is obtained if more constraints are added to the problem formulation.
15. Let f_n be the minimum value for the cost function with n design variables for a problem. If the number of design variables for the same problem is increased to, say $m = 2n$, then $f_m > f_n$ where f_m is the minimum value for the cost function with m design variables.

2.20* A trucking company wants to purchase several new trucks. It has \$2 million to spend. The investment should yield a maximum of trucking capacity for each day in tonnes × kilometers. Data for the three available truck models are given in Table E2-20; i.e., truck load capacity, average speed, crew required/shift, hours of

operations for three shifts, and the cost of each truck. There are some limitations on the operations that need to be considered. The labor market is such that the company can hire at the most 150 persons to operate the trucks. Garage and maintenance facilities can handle at the most 25 trucks. How many trucks of each type should the company purchase? Formulate the design optimization problem.

TABLE E2-20 Data for Available Trucks

<i>Truck model</i>	<i>Truck load capacity (tonnes)</i>	<i>Average truck speed (km/h)</i>	<i>Crew required per shift</i>	<i>No. of hours of operations per day (3 shifts)</i>	<i>Cost of each truck (\$)</i>
A	10	55	1	18	40,000
B	20	50	2	18	60,000
C	18	50	2	21	70,000

2.21* A large steel corporation has two iron ore reduction plants. Each plant processes iron ore into two different ingot stocks. They are shipped to any of the three fabricating plants where they are made into either of the two finished products. In total, there are two reduction plants, two ingot stocks, three fabricating plants, and two finished products.

For the coming season, the company wants to minimize total tonnage of iron ore processed in its reduction plants, subject to production and demand constraints. Formulate the design optimization problem and transcribe it into the standard model.

Nomenclature

- $a(r, s)$ tonnage yield of ingot stock s from 1 ton of iron ore processed at reduction plant r
- $b(s, f, p)$ total yield from 1 ton of ingot stock s shipped to fabricating plant f and manufactured into product p
- $c(r)$ iron ore processing capacity in tonnage at reduction plant r
- $k(f)$ capacity of the fabricating plant f in tonnage for all stocks
- $D(p)$ tonnage demand requirement for product p

Production and demand constraints

1. The total tonnage of iron ore processed by both reduction plants must equal the total tonnage processed into ingot stocks for shipment to the fabricating plants.
2. The total tonnage of iron ore processed by each reduction plant cannot exceed its capacity.
3. The total tonnage of ingot stock manufactured into products at each fabricating plant must equal the tonnage of ingot stock shipped to it by the reduction plants.
4. The total tonnage of ingot stock manufactured into products at each fabricating plant cannot exceed its available capacity.
5. The total tonnage of each product must equal its demand.

Constants for the problem

$a(1, 1) = 0.39$	$c(1) = 1,200,000$	$k(1) = 190,000$	$D(1) = 330,000$
$a(1, 2) = 0.46$	$c(2) = 1,000,000$	$k(2) = 240,000$	$D(2) = 125,000$
$a(2, 1) = 0.44$		$k(3) = 290,000$	
$a(2, 2) = 0.48$			
	$b(1, 1, 1) = 0.79$	$b(1, 1, 2) = 0.84$	
	$b(2, 1, 1) = 0.68$	$b(2, 1, 2) = 0.81$	
	$b(1, 2, 1) = 0.73$	$b(1, 2, 2) = 0.85$	
	$b(2, 2, 1) = 0.67$	$b(2, 2, 2) = 0.77$	
	$b(1, 3, 1) = 0.74$	$b(1, 3, 2) = 0.72$	
	$b(2, 3, 1) = 0.62$	$b(2, 3, 2) = 0.78$	

- 2.22 **Optimization of a water canal.** Design a water canal having a cross-sectional area of 150m^2 . Least construction costs occur when the volume of the excavated material equals the amount of material required for the dykes, as shown in Fig. E2-22. Formulate the problem to minimize the dugout material A_1 . Transcribe the problem into the standard design optimization model (created by V. K. Goel).

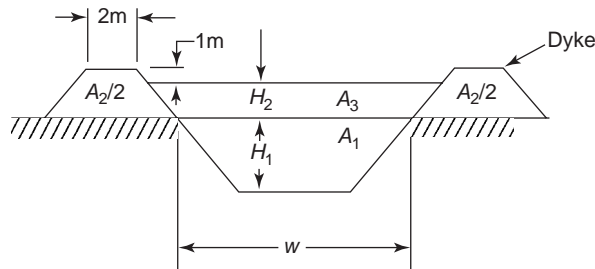


FIGURE E2-22 Cross section of a canal.

- 2.23 A cantilever beam is subjected to the point load P (kN), as shown in Fig. E2-23. The maximum bending moment in the beam is Pl (kN·m) and the maximum shear is P (kN). Formulate the minimum mass design problem using a hollow circular cross section. The material should not fail under bending stress or shear stress. The maximum bending stress is calculated as

$$\sigma = \frac{Pl}{I} R_o$$

where I = moment of inertia of the cross section. The maximum shearing stress is calculated as

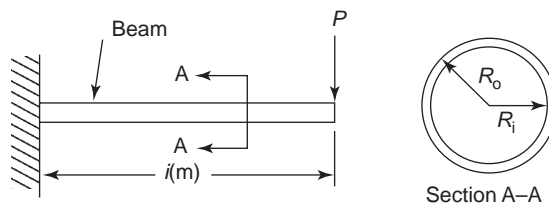


FIGURE E2-23 Cantilever beam.

$$\tau = \frac{P}{3I}(R_o^2 + R_o R_i + R_i^2)$$

Transcribe the problem into the standard design optimization model (also use $R_o \leq 40.0$ cm, $R_i \leq 40.0$ cm). Use the following data: $P = 14$ kN; $l = 10$ m; mass density, $\rho = 7850$ kg/m³; allowable bending stress, $\sigma_b = 165$ MPa; allowable shear stress, $\tau_a = 50$ MPa.

- 2.24 Design a hollow circular beam shown in Fig. E2-24 for two conditions: when $P = 50$ (kN), the axial stress σ should be less than σ_a , and when $P = 0$, deflection δ due to self-weight should satisfy $\delta \leq 0.001l$. The limits for dimensions are $t = 0.10$ to 1.0 cm, $R = 2.0$ to 20.0 cm, and $R/i \geq 20$. Formulate the minimum weight design problem and transcribe it into the standard form. Use the following data: $\delta = 5wl^4/384EI$; $w =$ self weight force/length (N/m); $\sigma_a = 250$ MPa; modulus of elasticity, $E = 210$ GPa; mass density, $\rho = 7800$ kg/m³; $\sigma = P/A$; gravitational constant, $g = 9.80$ m/s²; moment of inertia, $I = \pi R^3 t$ (m⁴).

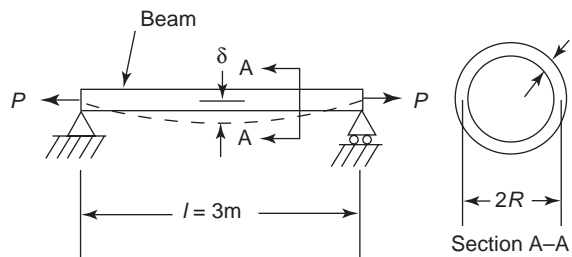


FIGURE E2-24 Hollow circular beam.

3 Graphical Optimization

Upon completion of this chapter, you will be able to:

- Graphically solve any optimization problem having two design variables
- Plot constraints and identify their feasible/infeasible side
- Identify the feasible region/feasible set for the problem
- Plot objective function contours through the feasible region
- Graphically locate the optimum solution for a problem and identify active/inactive constraints
- Identify problems that may have multiple, unbounded, or infeasible solutions

Optimization problems having only two design variables can be solved by observing the way they are graphically represented. All constraint functions are plotted, and a set of feasible designs (the feasible set) for the problem is identified. Objective function contours are then drawn and the optimum design is determined by visual inspection. In this chapter, we illustrate the graphical solution process and introduce several concepts related to optimum design problems. In the following section, a design optimization problem is formulated and used to describe the solution process. Some concepts related to design optimization problems are also described. Several more example problems are solved in later sections to illustrate the concepts and procedure.

3.1 Graphical Solution Process

3.1.1 Profit Maximization Problem

Step 1: Project/Problem Statement A company manufactures two machines, A and B. Using available resources, either 28 A or 14 B machines can be manufactured daily. The sales department can sell up to 14 A machines or 24 B machines. The shipping facility can handle no more than 16 machines per day. The company makes a profit of \$400 on each A machine and \$600 on each B machine. How many A and B machines should the company manufacture every day to maximize its profit?

Step 2: Data and Information Collection Defined in the project statement.

Step 3: Identification/Definition of Design Variables The following two design variables are identified in the problem statement:

x_1 = number of A machines manufactured each day
 x_2 = number of B machines manufactured each day

Step 4: Identification of a Criterion to Be Optimized The objective is to maximize daily profit, which can be expressed in terms of design variables as

$$P = 400x_1 + 600x_2 \quad (a)$$

Step 5: Identification of Constraints Design constraints are placed on manufacturing capacity, limitations on the sales personnel, and restrictions on the shipping and handling facility. The constraint on the shipping and handling facility is quite straightforward, expressed as

$$x_1 + x_2 \leq 16 \text{ (shipping and handling constraint)} \quad (b)$$

Constraints on manufacturing and sales facilities are a bit tricky. First, consider the manufacturing limitation. It is assumed that if the company is manufacturing x_1 A machines per day, then the remaining resources and equipment can be proportionately utilized to manufacture B number of machines, and vice versa. Therefore, noting that $x_1/28$ is the fraction of resources used to produce A machines and $x_2/14$ is the fraction used for B, the constraint is expressed as

$$\frac{x_1}{28} + \frac{x_2}{14} \leq 1 \text{ (manufacturing constraint)} \quad (c)$$

Similarly, the constraint on sales department resources is given as

$$\frac{x_1}{14} + \frac{x_2}{24} \leq 1 \text{ (limitation on sales department)} \quad (d)$$

Finally, the design variables must be nonnegative as

$$x_1, x_2 \geq 0 \quad (e)$$

Note that for this problem, the formulation remains valid even when a design variable has zero value. The problem has two design variables and five inequality constraints. All functions of the problem are linear in variables x_1 and x_2 . Therefore, it is a *linear programming problem*.

3.1.2 Step-by-Step Graphical Solution Procedure

Step 1: Coordinate System Set-up The first step in the solution process is to set up an origin for the x - y coordinate system and scales along the x and y axes. By looking at the constraint functions, a coordinate system for the profit maximization problem can be set up using a range of 0 to 25 along both the x and y axes. In some cases, the scale may need to be adjusted after the problem has been graphed because the original scale may provide too small or too large a graph for the problem.

Step 2: Inequality Constraint Boundary Plot To illustrate the graphing of a constraint, let us consider the inequality $x_1 + x_2 \leq 16$, given in Eq. (b). To represent the constraint graphically, we first need to plot the constraint boundary; i.e., plot the points that satisfy the constraint as an equality $x_1 + x_2 = 16$. This is a linear function of the variables x_1 and x_2 . To plot such a function, we need two points that satisfy the equation $x_1 + x_2 = 16$. Let these points be calculated as (16,0) and (0,16). Locating these points on the graph and joining them by a straight line produces the line F–J, as shown in Fig. 3-1. Line F–J then represents the boundary of the feasible region for the inequality constraint $x_1 + x_2 \leq 16$. Points on one side of this line will violate the constraint, while those on the other side will satisfy it.

Step 3: Identification of Feasible Region for an Inequality The next task is to determine which side of constraint boundary F–J is feasible for the constraint $x_1 + x_2 \leq 16$. To accomplish this task, we select a point on either side of F–J at which to evaluate the constraint. For example, at point (0,0), the left side of the constraint $x_1 + x_2 \leq 16$ has a value of 0. Because the value is less than 16, the constraint is satisfied and the region below F–J is feasible. We can test the constraint at another point on the opposite side of F–J, say at point (10,10). At this point the constraint is violated because the left side of the constraint function is 20, which is larger than 16. Therefore, the region above F–J is infeasible with respect to the constraint $x_1 + x_2 \leq 16$, as shown in Fig. 3-2. The infeasible region is “shaded-out” or “hatched-out,” a convention that is used throughout this text. Note that if this was an equality constraint $x_1 + x_2 = 16$, then the feasible region for the constraint would only be the points on line F–J. Although there is an infinite number of points on F–J, the feasible region for the equality constraint is much smaller than that for the same constraint written as an inequality.

Step 4: Identification of Feasible Region By following the procedure described in Step 3, all constraints are plotted on the graph and the feasible region for each constraint is identified. Note that the constraints $x_1, x_2 \geq 0$ restrict the feasible region to the first quadrant

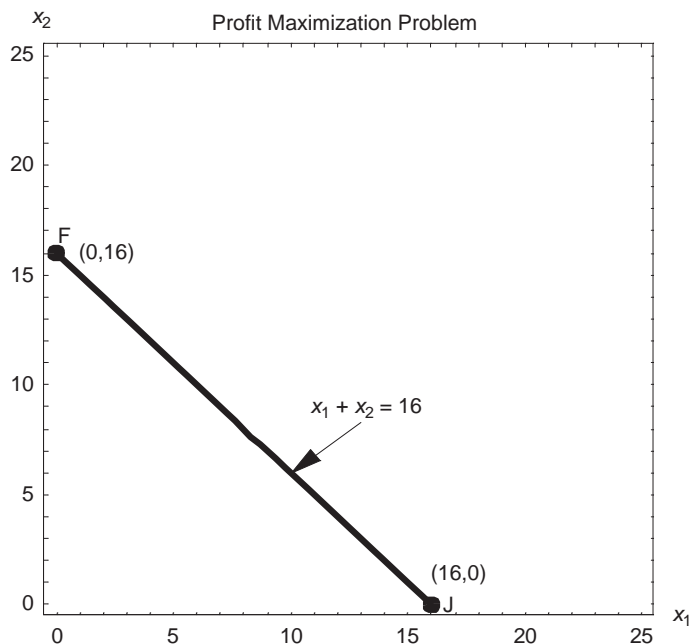


FIGURE 3-1 Constraint boundary for the inequality $x_1 + x_2 \leq 16$.

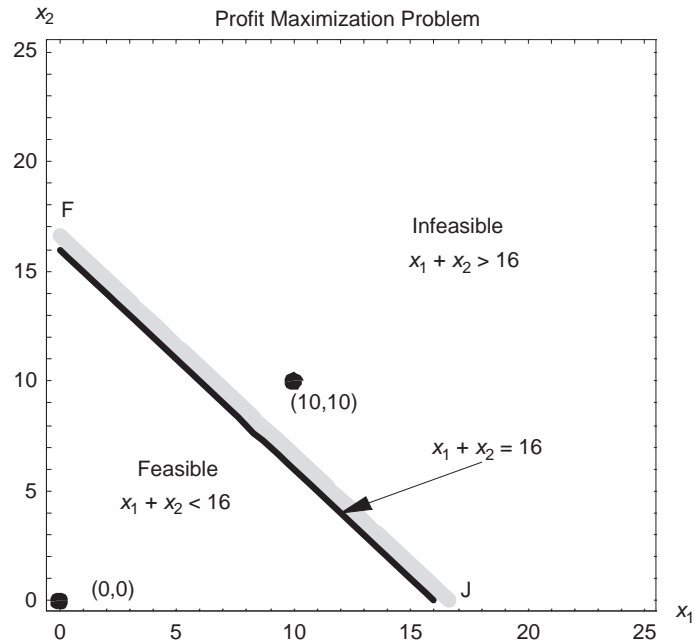


FIGURE 3-2 Feasible/infeasible side for the inequality $x_1 + x_2 \leq 16$.

of the coordinate system. The intersection of feasible regions for all constraints provides the feasible region for the profit maximization problem, indicated as ABCDE in Fig. 3-3. Any point in this region or on its boundary provides a feasible solution to the problem.

Step 5: Plotting Objective Function Contours The next task is to plot the objective function on the graph and locate its optimum points. For the present problem, the objective is to maximize the profit, $P = 400x_1 + 600x_2$, which involves three variables: P , x_1 , and x_2 . The function needs to be represented on the graph so that the value of P can be compared for different feasible designs and the best design can be located. However, because there is an infinite number of feasible points, it is not possible to evaluate the objective function at every point. One way of overcoming this impasse is to plot the contours of the objective function. A *contour* is a curve on the graph that connects all points having the same objective function value. A collection of points on a contour is also called the *level set*. If the objective function is to be minimized, the contours are also called *iso-cost curves*. To plot a contour through the feasible region, we need to assign it a value. To obtain this value, consider a point in the feasible region and evaluate the profit function there. For example, at point (6,4), P is $P = 6 \times 400 + 4 \times 600 = 4800$. To plot the $P = 4800$ contour, we plot the function $400x_1 + 600x_2 = 4800$. This contour is shown in Fig. 3-4.

Step 6: Identification of Optimum Solution To locate an optimum point for the objective function, we need at least two contours that pass through the feasible region. We can then observe trends for the values of the objective function at different feasible points to locate the best solution point. Contours for $P = 2400$, 4800 , and 7200 are plotted in Fig. 3-5. We now observe the following trend: as the contours move up toward point D, feasible designs can be found with larger values for P . It is clear from observation that point D has the largest value for P in the feasible region. We now simply read the coordinates of point D (4,12) to obtain the optimum design, having a maximum value for the profit function as $P = 8800$.

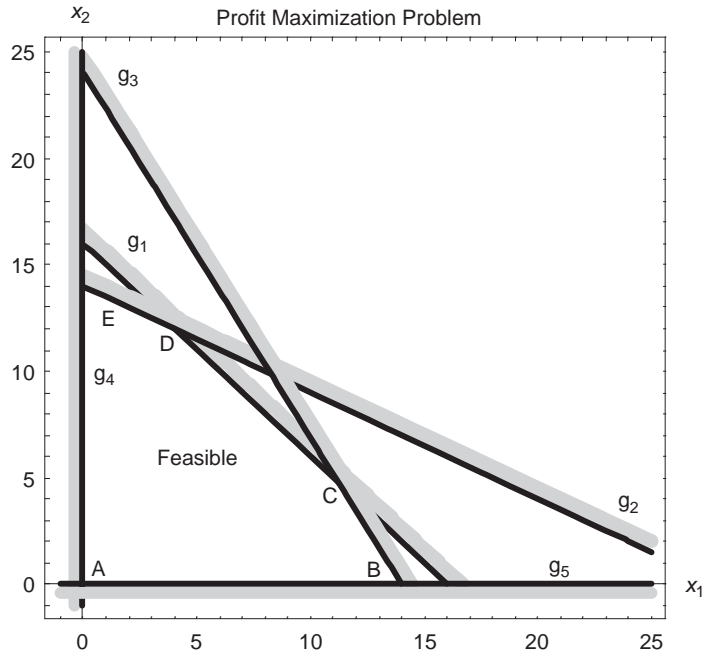


FIGURE 3-3 Feasible region for the profit maximization problem.

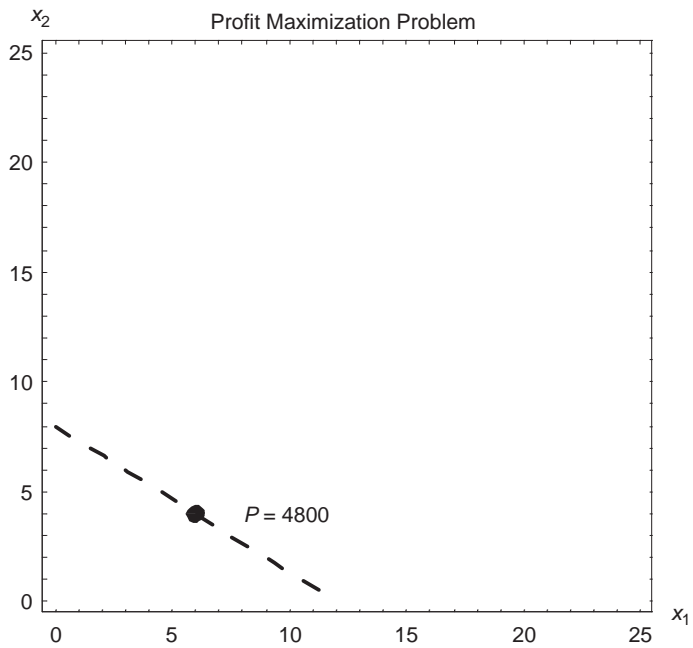


FIGURE 3-4 Plot of $P = 4800$ objective function contour for the profit maximization problem.

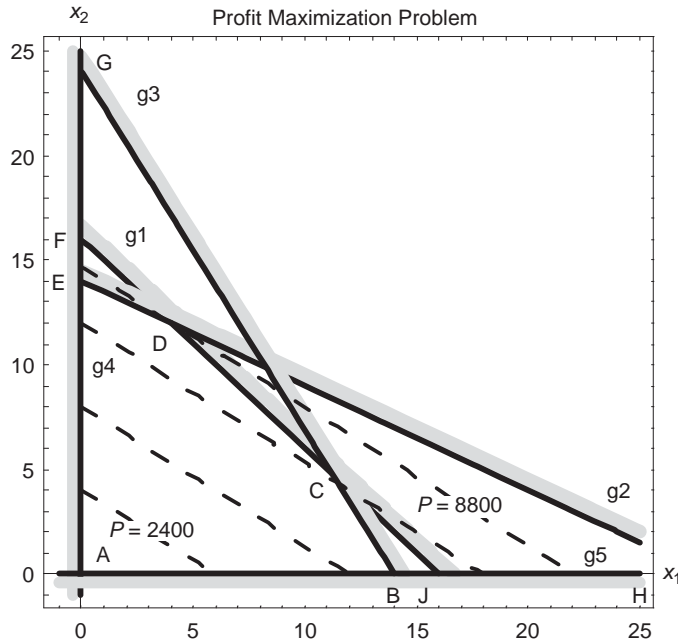


FIGURE 3-5 Graphical solution for the profit maximization problem. Optimum point $D = (4, 12)$. Maximum profit, $P = 8800$.

Thus, the best strategy for the company is to manufacture 4 A and 12 B machines to maximize its daily profit. The inequality constraints in Eqs. (b) and (c) are *active* at the optimum; i.e., they are satisfied at equality. These represent limitations on shipping and handling facilities, and manufacturing. The company can think about relaxing these constraints to improve its profit. All other inequalities are strictly satisfied, and therefore, *inactive*.

Note that in this example the design variables must have integer values. Fortunately, the optimum solution has integer values for the variables. If this were not the case, we would have used the procedure suggested in Section 2.11.4 or in Chapter 15 to solve this problem. Note also that for this example all functions are linear in design variables. Therefore, all curves in Figs. 3-1 through 3-5 are *straight lines*. In general, the functions of a design problem may not be linear, in which case curves must be plotted to identify the feasible region, and contours or iso-cost curves must be drawn to identify the optimum design. To *plot a non-linear function*, a table of numerical values for x_1 and x_2 must be generated for the function. These points must be then plotted on a graph and connected by a smooth curve.

3.2 Use of Mathematica for Graphical Optimization

It turns out that good programs, such as Mathematica, are available to implement the step-by-step procedure of the previous section and obtain a graphical solution for the problem on the computer screen. Mathematica is an interactive software package with many capabilities; however, we shall explain its use to solve a two-variable optimization problem by plotting all functions on the computer screen. Although other commands for plotting functions are available, the most convenient one for working with inequality constraints and objective function contours is the *ContourPlot* command. As with most Mathematica commands, this command is followed by what we call *subcommands* as “arguments” that define the nature

of the plot. All Mathematica commands are case sensitive so it is important to pay attention to which letters are capitalized.

Mathematica input is organized into what is called a “notebook.” A *notebook* is divided into *cells* with each cell containing input that can be executed independently. For explaining the graphical optimization capability of Mathematica5, we shall use the profit maximization problem of the previous section. Note that the commands used here may change in future releases of the program. We start by entering into the notebook the problem functions as

```
P=400*x1+600*x2;
g1=x1+x2-16; (*shipping and handling constraint*)
g2=x1/28+x2/14-1; (*manufacturing constraint*)
g3=x1/14+x2/24-1; (*limitation on sales department*)
g4=-x1;
g5=-x2;
```

This input illustrates some basic features concerning Mathematica format. Note that the ENTER key acts simply as a carriage return, taking the blinking cursor to the next line. Pressing SHIFT and ENTER actually inputs the typed information into Mathematica. When no immediate output from Mathematica is desired, the input line must end with a semicolon (;). If the semicolon is omitted, Mathematica will simplify the input and display it on the screen or execute an arithmetic expression and display the result. Comments are bracketed as (*Comment*). Note also that all the constraints are assumed to be in the standard “ \leq ” form. This helps in identifying the infeasible region for constraints on the screen using the *ContourPlot* command.

3.2.1 Plotting Functions

The Mathematica command used to plot the contour of a function, say $g_1 = 0$, is entered as

```
Plotg1=ContourPlot[g1,{x1,0,25},{x2,0,25}, ContourShading→False, Contours→{0},
ContourStyle→{{Thickness[.01]}}, Axes→True, AxesLabel→{"x1","x2"},
PlotLabel→"Profit Maximization Problem", Epilog→{Disk[{0,16},{.4,.4}],
Text["(0,16)",{2,16}], Disk[{16,0},{.4,.4}], Text["(16,0)",{17,1.5}],
Text["F",{0,17}], Text["J",{17,0}], Text["x1+x2=16",{13,9}],
Arrow[{13,8.3},{10,6}]}, DefaultFont→{"Times",12}, ImageSize→72 5];
```

Plotg1 is simply an arbitrary name referring to the data points for the function g_1 determined by the *ContourPlot* command; it is used in future commands to refer to this particular plot. This *ContourPlot* command plots a contour defined by the equation $g_1 = 0$ as in Fig. 3-1. Arguments of the *ContourPlot* command containing various subcommands are explained as follows (note that the arguments are separated by commas and are enclosed in square brackets []):

g1: function to be plotted.

{*x1*, 0, 25}, {*x2*, 0, 25}: ranges for the variables *x1* and *x2*; 0 to 25.

ContourShading → False: indicates that shading will not be used to plot contours, whereas *ContourShading* → True would indicate that shading will be used (note that most subcommands are followed by an arrow “→” or “->” and a set of parameters enclosed in braces {}).

Contours → {0}: contour values for g_1 , one contour is requested having 0 value.

ContourStyle → {{Thickness[.01]}}, defines characteristics of the contour such as thickness and color. Here, the thickness of the contour is specified as “.01”. It is

given as a fraction of the total width of the graph and needs to be determined by trial and error.

Axes → True: indicates whether axes should be drawn at the origin; in the present case, where the origin (0, 0) is located at the bottom left corner of the graph, the *Axes* subcommand is irrelevant except that it allows for the use of the *AxesLabel* command.

AxesLabel → {"x1","x2"}: allows one to indicate labels for each axis.

PlotLabel → "Profit Maximization Problem": puts a label at the top of the graph.

Epilog → { . . . }: allows insertion of additional graphics primitives and text in the figure on the screen; Disk [{0,16}, {.4,.4}] allows insertion of a dot at the location (0,16) of radius .4 in both directions; Text [{"(0,16)", (2,16)}] allows "(0,16)" to be placed at the location (2,16).

ImageSize → 72 5: indicates that the width of the plot should be 5 inches; the size of the plot also can be adjusted by selecting the image in Mathematica and dragging one of the black square control points; the images in Mathematica can be copied and pasted to a word processor file.

DefaultFont → {"Times",12}: specifies the preferred font and size for the text.

3.2.2 Identification and Hatching of Infeasible Region for an Inequality

Figure 3-2 is created using a slightly modified *ContourPlot* command used earlier for Fig. 3-1:

```
Plotg1=ContourPlot[g1,{x1,0,25},{x2,0,25}, ContourShading→False,Contours→{0,.65},
ContourStyle→{{Thickness[.01]},{GrayLevel[.8],Thickness[.025]}}], Axes→True,
AxesLabel→{"x1","x2"}, PlotLabel→"Profit Maximization Problem",
Epilog→{Disk[{10,10},{.4,.4}], Text["(10,10)",{11,9}], Disk[{0,0},{.4,.4}],
Text["(0,0)",{2,.5}], Text["x1+x2=16",{18,7}], Arrow[{18,6.3},{12,4}],
Text["Infeasible",{17,17}], Text["x1+x2>16",{17,15.5}], Text["Feasible",{5,6}],
Text["x1+x2<16",{5,4.5}]}, DefaultFont→{"Times",12}, ImageSize→72 5];
```

Here, two contour lines are specified, the second one having a small positive value. This is indicated by the command: *Contours* → {0, .65}. The constraint boundary is represented by the contour $g_1 = 0$. The contour $g_1 = 0.65$ will pass through the infeasible region, where the positive number 0.65 is determined by trial and error. To shade the infeasible region, the characteristics of the contour are changed. Each set of brackets {} with the *ContourStyle* subcommand corresponds to a specific contour. In this case, {Thickness[.01]} provides characteristics for the first contour $g_1 = 0$, and {GrayLevel[.8],Thickness[0.025]} provides characteristics for the second contour $g_1 = 0.65$. GrayLevel specifies a color for the contour line. A gray level of 0 yields a black line, whereas a gray level of 1 yields a white line. Thus, this *ContourPlot* command essentially draws one thin, black line and one thick, gray line. This way the infeasible side of an inequality is shaded out.

3.2.3 Identification of Feasible Region

By using the foregoing procedure, all constraint functions for the problem are plotted and their feasible sides are identified. The plot functions for the five constraints g_1 to g_5 are named Plotg1, Plotg2, Plotg3, Plotg4, Plotg5. All these functions are quite similar to the one that was created using the *ContourPlot* command explained earlier. As an example, Plotg4 function is given as

```
Plotg4=ContourPlot[g4,{x1,-1,25},{x2,-1,25}, ContourShading→False,
Contours→{0,.35}, ContourStyle→{{Thickness[.01]},{GrayLevel[.8],Thickness[.02]}}],
DisplayFunction→Identity];
```

The *DisplayFunction* \rightarrow *Identity* subcommand is added to the *ContourPlot* command to suppress display of output from each *Plotg* function; without that Mathematica executes each *Plotg* function and displays the results. Next, with the following *Show* command, the five plots are combined to display the complete feasible set in Fig. 3-3:

```
Show[{Plotg1,Plotg2,Plotg3,Plotg4,Plotg5}, Axes→True,AxesLabel→{"x1","x2"},
PlotLabel→"Profit Maximization Problem", DefaultFont→{"Times",12}, Epilog→
{Text["g1",{2.5,16.2}], Text["g2",{24,4}], Text["g3",{2,24}], Text["g5",{21,1}],
Text["g4",{1,10}], Text["Feasible",{5,6}]}, DefaultFont→{"Times",12}, ImageSize→72
5,DisplayFunction → $DisplayFunction];
```

The *Text* subcommands are included to add text to the graph at various locations. The *DisplayFunction* \rightarrow *\$DisplayFunction* subcommand is added to display the final graph; without that it is not displayed.

3.2.4 Plotting of Objective Function Contours

The next task is to plot the objective function contours and locate its optimum point. The objective function contours of values 2400, 4800, 7200, 8800, shown in Fig. 3-4 are drawn by using the *ContourPlot* command as follows:

```
PlotP=ContourPlot[P,{x1,0,25},{x2,0,25}, ContourShading→False, Contours→{4800},
ContourStyle→{{Dashing[{.03,.04}], Thickness[.007]}}, Axes→True,
AxesLabel→{"x1","x2"}, PlotLabel→"Profit Maximization Problem",
DefaultFont→{"Times",12}, Epilog→{Disk[{6,4},{.4,.4}], Text["P= 4800",{9.75,4}]},
ImageSize→72 5];
```

The *ContourStyle* subcommand provides four sets of characteristics, one for each contour. *Dashing*[{a,b}] yields a dashed line with "a" as the length of each dash and "b" as the space between dashes. These parameters represent a fraction of the total width of the graph.

3.2.5 Identification of Optimum Solution

The *Show* command used to plot the feasible region for the problem in Fig. 3-3 can be extended to plot the profit function contours as well. Figure 3-5 contains the graphical representation for the problem obtained using the following *Show* command:

```
Show[{Plotg1,Plotg2,Plotg3,Plotg4,Plotg5, PlotP}, Axes→True, AxesLabel→{"x1","x2"},
PlotLabel → "Profit Maximization Problem", DefaultFont→{"Times",12},
Epilog→{Text["g1",{2.5,16.2}], Text["g2",{24,4}], Text["g3",{3,23}], Text["g5",{23,1}],
Text["g4",{1,10}], Text["P= 2400",{3.5,2}], Text["P= 8800",{17,3.5}], Text["G",{1,24.5}],
Text["C",{10.5,4}], Text["D",{3.5,11}], Text["A",{1,1}], Text["B",{14,-1}],Text["J",{16,-1}],
Text["H",{25,-1}], Text["E",{-1,14}], Text["F",{-1,16}]}, DefaultFont→{"Times",12},
ImageSize→72 5, DisplayFunction →$DisplayFunction];
```

Additional *Text* subcommands have been added to label different objective function contours and different points. The final graph is used to obtain the graphical solution. The *Disk* subcommand can be added to the *Epilog* command to put a dot at the optimum point.

3.3 Use of MATLAB for Graphical Optimization

MATLAB is another software package that has many capabilities to solve engineering problems. For example, it can be used to plot problem functions and to solve graphically a two-variable optimization problem. In this section, we shall explain use of the program for this purpose; other uses of the program for solving optimization problems are explained in Chapter 12. There are two modes of input with MATLAB. One may enter commands interactively, one at a time, and results are displayed immediately after each command. Alternatively, one may create an input file, called an *M-file*, that is executed in batch mode. The M-file can be created using the text editor in MATLAB. To access this editor, select “File”, “New”, and “M-file”. When saved, this file will have a suffix of “.m.” To submit or run the file, after starting MATLAB, simply type the name of the file you wish to run, without the suffix “.m” (the *current directory* must be the directory where the file is located). In this section, we shall solve the profit maximization problem of previous sections using MATLAB6.5. It is important to note with future releases, the commands discussed below may change.

3.3.1 Plotting of Function Contours

For contour plots, the first command in the input file is entered as follows:

```
[x1,x2]=meshgrid(-1.0:0.5:25.0, -1.0:0.5:25.0);
```

This command creates a grid or array of points where all functions to be plotted are evaluated. The command indicates that x_1 and x_2 will start at -1.0 and increase in increments of 0.5 up to 25.0 . These variables now represent two-dimensional arrays and require special attention in operations with them. “*” and “/” indicate scalar multiplication and division respectively, whereas “.*” and “./” indicate element-by-element multiplication and division. “.^” is used to apply an exponent to each element of a vector or a matrix. The semicolon “;” after a command prevents MATLAB from displaying the numerical results immediately, i.e., all of the values for x_1 and x_2 . This use of a semicolon is a MATLAB convention for most commands. The “contour” command is used for plotting all problem functions on the screen. The “.m file” for the profit maximization problem with explanatory comments is prepared and displayed in Table 3-1. Note that the comments in the “.m file” are preceded by the percent sign, %. The comments are ignored during MATLAB execution. Also note that matrix division and multiplication capabilities are not used in the present example as the variables in the problem functions are only multiplied or divided by a scalar rather than another variable. If, for instance, a term such as x_1x_2 was present, then the element-by-element operation $x_1.*x_2$ would be necessary.

The procedure used to identify the infeasible side of an inequality is the same as explained in the previous section. Two contours are plotted for the inequality; one of value 0 and the other of small positive value. The second contour will pass through the infeasible region for the problem. The thickness of the infeasible contour is changed to indicate the infeasible side of the inequality using the graph editing capability that is explained in the next section. This way all the constraint functions are plotted and the feasible region for the problem is identified. By observing the trend of the objective function contours, we can identify the optimum point for the problem.

3.3.2 Editing of Graph

Once the graph has been created using the previous commands, it is possible to edit it before printing it or copying it to a text editor. In particular, one may need to modify the appearance of the infeasible contours of the constraints and edit text in the graph. To do this, first select “Current Object Properties . . .” under the “Edit” tab on the graph window. Then, double click any item in the graph to edit its properties. For instance, one may increase the thickness of the infeasible contours to hatch out the infeasible region. In addition, text may

TABLE 3-1 MATLAB File for Profit Maximization Problem

```
%Create a grid from -1 to 25 with an increment of 0.5 for the variables x1 and x2
[x1,x2]=meshgrid(-1:0.5:25.0,-1:0.5:25.0);
%Enter functions for the profit maximization problem
f=400*x1+600*x2;
g1=x1+x2-16;
g2=x1/28+x2/14-1;
g3=x1/14+x2/24-1;
g4=-x1;
g5=-x2;
%Initialization statements; these need not end with a semicolon
    cla reset
    axis auto
                                %Minimum and maximum values for axes are
                                %determined automatically
                                %Limits for x- and y-axes may also be specified with
                                %the command
                                %axis ([xmin xmax ymin ymax])
    xlabel('x1'),ylabel('x2')
                                %Specifies labels for x- and y-axes
    title ('Profit Maximization Problem')
                                %Displays a title for the problem
    hold on
                                %retains the current plot and axes properties for all
                                %subsequent plots
%Use the "contour" command to plot constraint and cost functions
    cv1=[0 .5];
                                %Specifies two contour values
    const1=contour(x1,x2,g1,cv1,'k');
                                %Plots two specified contours of g1; k = black color
    clabel(const1)
                                %Automatically puts the contour value on the graph
    text(1,16,'g1')
                                %Writes g1 at the location (1, 16)
    cv2=[0 .03];
    const2=contour(x1,x2,g2,cv2,'k');
    clabel(const2)
    text(23,3,'g2')
    const3=contour(x1,x2,g3,cv2,'k');
    clabel(const3)
    text(1,23,'g3')
    cv3=[0 .5];
    const4=contour(x1,x2,g4,cv3,'k');
    clabel(const4)
    text(.25,20,'g4')
    const5=contour(x1,x2,g5,cv3,'k');
    clabel(const5)
    text(19,.5,'g5')
    text(1.5,7,'Feasible Region')
    fv=[2400, 4800, 7200, 8800];
                                %Defines 4 contours for the profit function
    fs=contour(x1,x2,f,fv,'k-');
                                %'k-' specifies black dashed lines for profit function
                                %contours
    clabel(fs)
    hold off
                                %Indicates end of this plotting sequence
                                %Subsequent plots will appear in separate windows
```

be added, deleted, or moved as desired. Note that if MATLAB is rerun, any changes made directly to the graph are lost. For this reason, it is a good idea to save the graph as a ".fig" file, which then may be recalled with MATLAB. There are two ways for transferring the

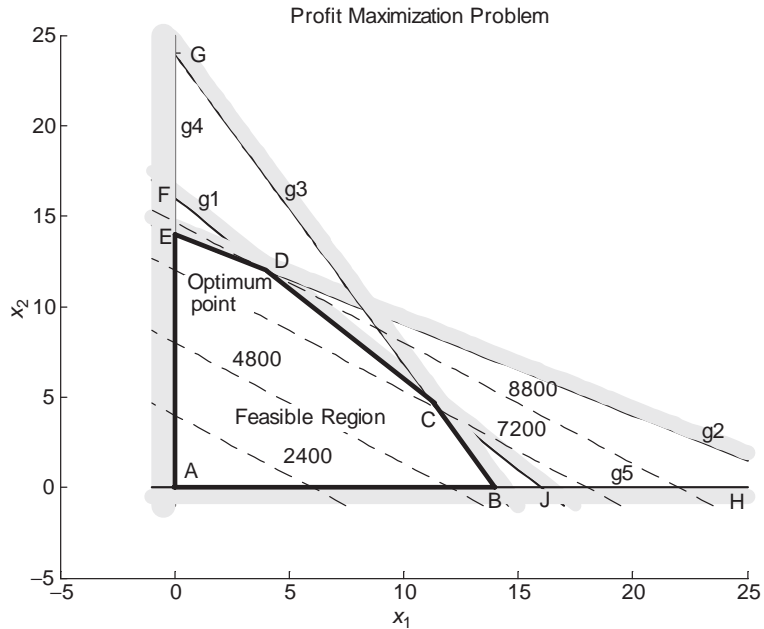


FIGURE 3-6 Graphical representation for the profit maximization problem with MATLAB.

graph to another document. First, select “Copy Figure” under the “Edit” tab. The figure then can be pasted as a bitmap into another document. Alternatively, one may select “Export . . .” under the “File” tab. The figure is exported as the specified file type and then can be inserted into another document through the “Insert” command. The final graph with MATLAB for the profit maximization problem is shown in Fig. 3-6.

3.4 Design Problem with Multiple Solutions

A situation can arise in which a constraint is parallel to the cost function. If the constraint is active at the optimum, then there are multiple solutions to the problem. To illustrate this situation, consider the following design problem: minimize $f(\mathbf{x}) = -x_1 - 0.5x_2$ subject to four inequality constraints

$$2x_1 + 3x_2 \leq 12, \quad 2x_1 + x_2 \leq 8, \quad -x_1 \leq 0, \quad -x_2 \leq 0$$

In this problem, the second constraint is parallel to the cost function. Therefore, there is a possibility of *multiple optimum designs*. Figure 3-7 provides a graphical solution to the problem. It can be seen that any point on the line B–C gives an optimum design. Thus the problem has infinite optimum solutions.

3.5 Problem with Unbounded Solution

Some design problems may not have a bounded solution. This situation can arise if we forget a constraint or incorrectly formulate the problem. To illustrate such a situation, consider the

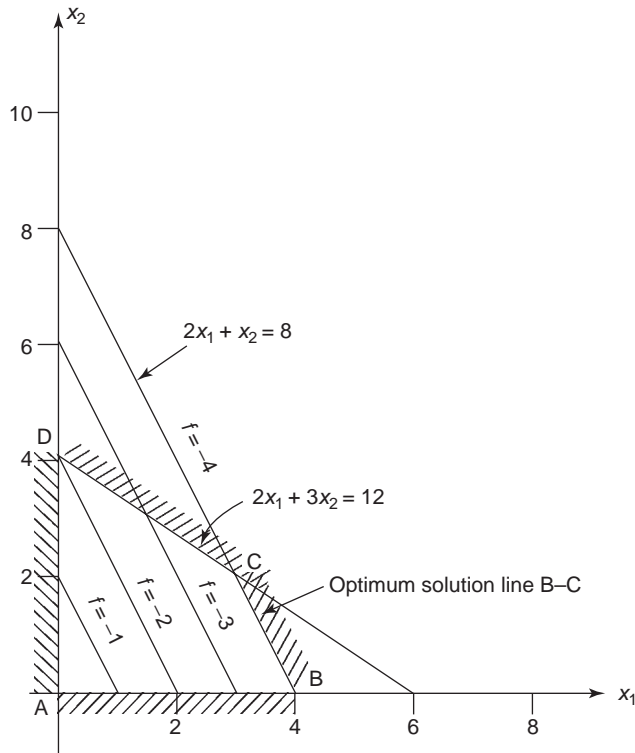


FIGURE 3-7 Example problem with multiple solutions.

following design problem: minimize $f(\mathbf{x}) = -x_1 + 2x_2$ subject to four inequality constraints

$$-2x_1 + x_2 \leq 0, \quad -2x_1 + 3x_2 \leq 6, \quad -x_1 \leq 0, \quad -x_2 \leq 0$$

The feasible set for the problem is shown in Fig. 3-8. Several cost function contours are shown. It can be seen that the feasible set is unbounded. Therefore, there is no finite optimum solution. We must re-examine the way the problem was formulated to correct the situation. It can be seen in Fig. 3-8 that the problem is under-constrained.

3.6 Infeasible Problem

If we are not careful in formulating a design problem, it may not have a solution, which happens when there are conflicting requirements or inconsistent constraint equations. There may also be no solution when we put *too many constraints* on the system, i.e., the constraints are so restrictive that no feasible solution is possible. These are called *infeasible problems*. To illustrate such a situation, consider the following problem: minimize $f(\mathbf{x}) = x_1 + 2x_2$ subject to six inequality constraints

$$3x_1 + 2x_2 \leq 6, \quad 2x_1 + 3x_2 \geq 12, \quad x_1, x_2 \leq 5, \quad x_1, x_2 \geq 0$$

Constraints for the problem are plotted in Fig. 3-9. It can be seen that there is no region within the design space that satisfies all constraints. Thus, the problem is infeasible. Basi-

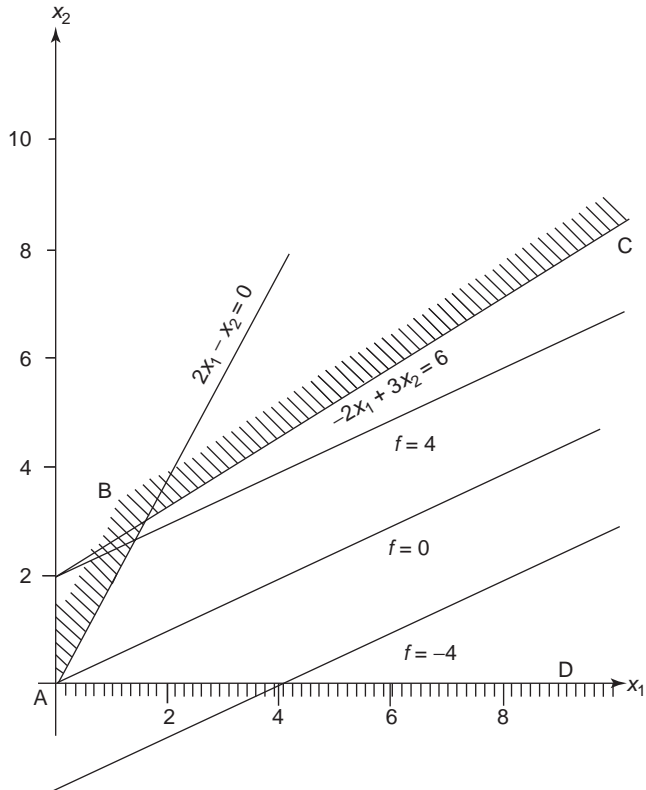


FIGURE 3-8 Example problem with unbounded solution.

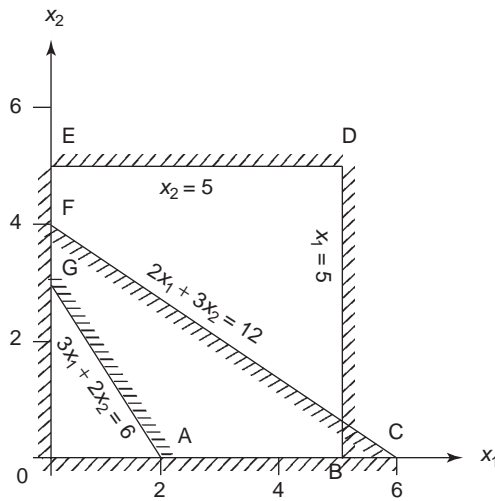


FIGURE 3-9 Example of infeasible design optimization problem.

cally, the first two constraints impose conflicting requirements on the design problem. The first requires the feasible design to be below the line A–G, whereas the second requires it to be above the line C–F. Since the two lines do not intersect in the first quadrant, there is no feasible region for the problem.

3.7 Graphical Solution for Minimum Weight Tubular Column

The design problem formulated in Section 2.7 will now be solved using the graphical method, with the following specifications: $P = 10\text{MN}$, $E = 207\text{GPa}$, $\rho = 7833\text{kg/m}^3$, $l = 5.0\text{m}$, and $\sigma_a = 248\text{MPa}$. Using this data, Formulation 1 for the problem is defined as: find mean radius R (m) and thickness t (m) to minimize the mass function:

$$f(R, t) = 2\rho l\pi Rt = 2(7833)(5)\pi Rt = 2.4608 \times 10^5 Rt, \text{ kg} \quad (\text{a})$$

subject to the four inequality constraints

$$g_1(R, t) = \frac{P}{2\pi Rt} - \sigma_a = \frac{10 \times 10^6}{2\pi Rt} - 248 \times 10^6 \leq 0 \quad (\text{stress constraint}) \quad (\text{b})$$

$$g_2(R, t) = P - \frac{\pi^3 ER^3 t}{4l^2} = 10 \times 10^6 - \frac{\pi^3 (207 \times 10^9) R^3 t}{4(5)(5)} \leq 0 \quad (\text{buckling load constraint}) \quad (\text{c})$$

$$g_3(R, t) = -R \leq 0; \quad (\text{d})$$

$$g_4(R, t) = -t \leq 0 \quad (\text{e})$$

Note that the explicit bound constraints are simply replaced by the nonnegativity constraints g_3 and g_4 . The constraints for the problem are plotted in Fig. 3-10 and the feasible region is indicated. Cost function contours for $f = 1000, 1500, 1579\text{kg}$ are also shown. Note that in this example the cost function contours run parallel to the stress constraint g_1 . Since g_1 is active at the optimum, the problem has an infinite number of optimum designs, i.e., the entire curve A–B in Fig. 3-10. We can read the coordinates of any point on the curve A–B as an optimum solution. In particular, point A, where constraints g_1 and g_2 intersect, is also an optimum point where $R^* = 0.1575\text{m}$ and $t^* = 0.0405\text{m}$. Note that the superscript * on a variable indicates its optimum value, a notation that will be used throughout this text.

Note also that this problem has nonlinear functions. To plot them, we generate tables of data points t versus R and connect them using smooth curves on the graph. For example, to plot the constraint boundary for g_2 ($R^3 t = 1.558 \times 10^{-4}$), we select values for t as 0.015, 0.03, 0.06, 0.075, 0.09, and calculate the values for R from $g_2 = 0$ as 0.218, 0.173, 0.1374, 0.1275, and 0.12. This procedure can be used to plot any nonlinear function of two variables. Figure 3-10 was generated using MATLAB with manual hatching of the infeasible region.

3.8 Graphical Solution for a Beam Design Problem

Step 1: Project/Problem Statement A beam of rectangular cross section is subjected to a bending moment of M (N·m) and a maximum shear force of V (N). The bending stress in the beam is calculated as $\sigma = 6M/bd^2$ (Pa) and average shear stress is calculated as $\tau = 3V/2bd$ (Pa), where b is the width and d is the depth of the beam. The allowable stresses in bending and shear are 10MPa and 2MPa, respectively. It is also desirable that the depth of the beam not exceed twice its width and that the cross-sectional area of the beam is minimized. In this section, we formulate and solve the problem using the graphical method.

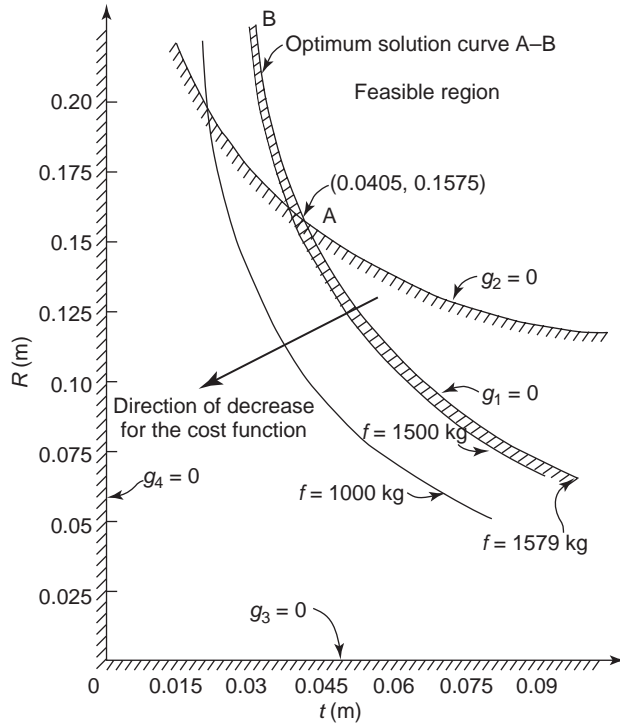


FIGURE 3-10 Graphical solution for a minimum weight tubular column.

Step 2: Data and Information Collection Let the bending moment $M = 40 \text{ kN}\cdot\text{m}$ and the shear force $V = 150 \text{ kN}$. All other data and necessary equations are given in the project statement. We shall formulate the problem using a consistent set of units as N and mm.

Step 3: Identification/Definition of Design Variables Two design variables are:

- d = depth of the beam, mm
- b = width of the beam, mm

Step 4: Identification of a Criterion to Be Optimized The cost function for the problem is the cross-sectional area, which is expressed as

$$f(b, d) = bd \tag{a}$$

Step 5: Identification of Constraints Constraints for the problem consist of bending stress, shear stress, and depth-to-width ratio. Bending and shear stresses are calculated as

$$\sigma = \frac{6M}{bd^2} = \frac{6(40)(1000)(1000)}{bd^2}, \text{ N/mm}^2 \tag{b}$$

$$\tau = \frac{3V}{2bd} = \frac{3(150)(1000)}{2bd}, \text{ N/mm}^2 \tag{c}$$

Allowable bending stress σ_a and allowable shear stress τ_a are given as

$$\sigma_a = 10 \text{ MPa} = 10 \times 10^6 \text{ N/m}^2 = 10 \text{ N/mm}^2 \quad (d)$$

$$\tau_a = 2 \text{ MPa} = 2 \times 10^6 \text{ N/m}^2 = 2 \text{ N/mm}^2 \quad (e)$$

Using Eqs. (b) through (e), we obtain the bending and shear stress constraints as

$$g_1 = \frac{6(40)(1000)(1000)}{bd^2} - 10 \leq 0 \text{ (bending stress)} \quad (f)$$

$$g_2 = \frac{3(150)(1000)}{2bd} - 2 \leq 0 \text{ (shear stress)} \quad (g)$$

The constraint that requires that the depth be no more than twice the width can be expressed as

$$g_3 = d - 2b \leq 0 \quad (h)$$

Finally, both design variables should be nonnegative:

$$g_4 = -b \leq 0; \quad g_5 = -d \leq 0 \quad (i)$$

In reality, b and d cannot both have zero value, so we should use some minimum value as lower bounds on them, i.e., $b \geq b_{\min}$ and $d \geq d_{\min}$.

Graphical Solution. Using MATLAB, the constraints for the problem are plotted in Fig. 3-11 and the feasible region is identified. Note that the cost function is parallel to the constraint g_2 (both functions have the same form: $bd = \text{constant}$). Therefore any point along the curve A–B represents an optimum solution. Thus, there is an infinite number of optimum designs. This is a desirable situation since a wide choice of optimum solutions is available to meet a designer's needs.

The optimum cross-sectional area is $112,500 \text{ mm}^2$. Point B corresponds to an optimum design of $b = 237 \text{ mm}$ and $d = 474 \text{ mm}$. Point A corresponds to $b = 527.3 \text{ mm}$ and $d =$

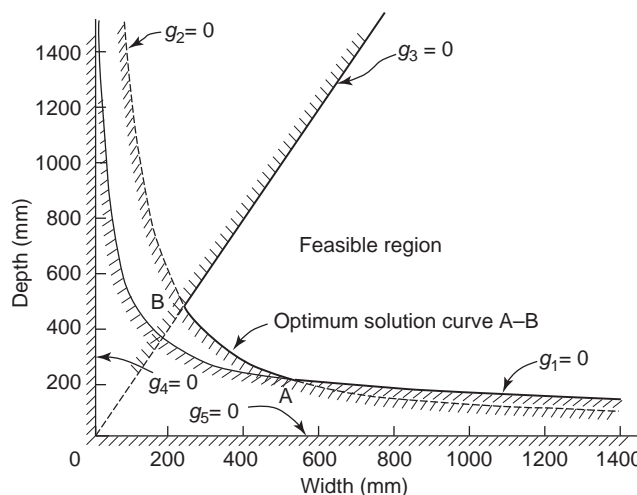


FIGURE 3-11 Graphical solution of the minimum area beam design problem.

213.3 mm. These points represent the two extreme optimum solutions; all other solutions lie between these two points on the curve A–B.

Exercises for Chapter 3

Solve the following problems using the graphical method.

- 3.1 Minimize $f(x_1, x_2) = (x_1 - 3)^2 + (x_2 - 3)^2$
subject to $x_1 + x_2 \leq 4$
 $x_1, x_2 \geq 0$
- 3.2 Maximize $F(x_1, x_2) = x_1 + 2x_2$
subject to $2x_1 + x_2 \leq 4$
 $x_1, x_2 \geq 0$
- 3.3 Minimize $f(x_1, x_2) = x_1 + 3x_2$
subject to $x_1 + 4x_2 \geq 48$
 $5x_1 + x_2 \geq 50$
 $x_1, x_2 \geq 0$
- 3.4 Maximize $F(x_1, x_2) = x_1 + x_2 + 2x_3$
subject to $1 \leq x_1 \leq 4$
 $3x_2 - 2x_3 = 6$
 $-1 \leq x_3 \leq 2$
 $x_2 \geq 0$
- 3.5 Maximize $F(x_1, x_2) = 4x_1x_2$
subject to $x_1 + x_2 \leq 20$
 $x_2 - x_1 \leq 10$
 $x_1, x_2 \geq 0$
- 3.6 Minimize $f(x_1, x_2) = 5x_1 + 10x_2$
subject to $10x_1 + 5x_2 \leq 50$
 $5x_1 - 5x_2 \geq -20$
 $x_1, x_2 \geq 0$
- 3.7 Minimize $f(x_1, x_2) = 3x_1 + x_2$
subject to $2x_1 + 4x_2 \leq 21$
 $5x_1 + 3x_2 \leq 18$
 $x_1, x_2 \geq 0$
- 3.8 Minimize $f(x_1, x_2) = x_1^2 - 2x_2^2 - 4x_1$
subject to $x_1 + x_2 \leq 6$
 $x_2 \leq 3$
 $x_1, x_2 \geq 0$
- 3.9 Minimize $f(x_1, x_2) = x_1x_2$
subject to $x_1 + x_2^2 \leq 0$
 $x_1^2 + x_2^2 \leq 9$
- 3.10 Minimize $f(x_1, x_2) = 3x_1 + 6x_2$
subject to $-3x_1 + 3x_2 \leq 2$
 $4x_1 + 2x_2 \leq 4$
 $-x_1 + 3x_2 \geq 1$

Develop an appropriate graphical representation for the following problems and determine all the local minimum and local maximum points.

- 3.11 $f(x, y) = x^2 + y^2$
subject to $y - x \leq 0$
 $x^2 + y^2 - 1 = 0$
- 3.12 $f(x, y) = 4x^2 + 3y^2 - 5xy - 8x$
subject to $x + y = 4$
- 3.13 $f(x, y) = 9x^2 + 13y^2 + 18xy - 4$
subject to $x^2 + y^2 + 2x = 16$
- 3.14 $f(x, y) = 2x + 3y - x^3 - 2y^2$
subject to $x + 3y \leq 6$
 $5x + 2y \leq 10$
 $x, y \geq 0$
- 3.15 $f(r, t) = (r - 8)^2 + (t - 8)^2$
subject to $12 \geq r + t$
 $t \leq 5$
 $r, t \geq 0$
- 3.16 $f(x_1, x_2) = x_1^3 - 16x_1 + 2x_2 - 3x_2^2$
subject to $x_1 + x_2 \leq 3$
- 3.17 $f(x, y) = 9x^2 + 13y^2 + 18xy - 4$
subject to $x^2 + y^2 + 2x \geq 16$
- 3.18 $f(r, t) = (r - 4)^2 + (t - 4)^2$
subject to $10 - r - t \geq 0$
 $5 \geq r$
 $r, t \geq 0$
- 3.19 $f(x, y) = -x + 2y$
subject to $-x^2 + 6x + 3y \leq 27$
 $18x - y^2 + 6x \geq 180$
 $x, y \geq 0$
- 3.20 $f(x_1, x_2) = (x_1 - 4)^2 + (x_2 - 2)^2$
subject to $10 \geq x_1 + 2x_2$
 $0 \leq x_1 \leq 3$
 $x_2 \geq 0$
- 3.21 Solve the rectangular beam problem of Exercise 2.17 graphically for the following data: $M = 80 \text{ kN}\cdot\text{m}$, $V = 150 \text{ kN}$, $\sigma_a = 8 \text{ MPa}$, and $\tau_a = 3 \text{ MPa}$.
- 3.22 Solve the cantilever beam problem of Exercise 2.23 graphically for the following data: $P = 10 \text{ kN}$; $l = 5.0 \text{ m}$; modulus of elasticity, $E = 210 \text{ GPa}$; allowable bending stress, $\sigma_a = 250 \text{ MPa}$; allowable shear stress, $\tau_a = 90 \text{ MPa}$; mass density, $\rho = 7850 \text{ kg/m}^3$; $R_o \leq 20.0 \text{ cm}$; $R_i \leq 20.0 \text{ cm}$.
- 3.23 For the minimum mass tubular column design problem formulated in Section 2.7, consider the following data: $P = 50 \text{ kN}$; $l = 5.0 \text{ m}$; modulus of elasticity, $E = 210 \text{ GPa}$; allowable stress, $\sigma_a = 250 \text{ MPa}$; mass density, $\rho = 7850 \text{ kg/m}^3$.
Treating mean radius R and wall thickness t as design variables, solve the design problem graphically imposing an additional constraint $R/t \leq 50$. This constraint is

needed to avoid local crippling of the column. Also impose the member size constraints as

$$0.01 \leq R \leq 1.0 \text{ m}; \quad 5 \leq t \leq 200 \text{ mm}$$

- 3.24 For Exercise 3.23, treat outer radius R_o and inner radius R_i as design variables, and solve the design problem graphically. Impose the same constraints as in Exercise 3.23.
- 3.25 Formulate the minimum mass column design problem of Section 2.7 using a hollow square cross section with outside dimension w and thickness t as design variables. Solve the problem graphically using the constraints and the data given in Exercise 3.23.
- 3.26 Consider the symmetric (members are identical) case of the two-bar truss problem discussed in Section 2.5 with the following data: $W = 10 \text{ kN}$; $\theta = 30^\circ$; height $h = 1.0 \text{ m}$; span $s = 1.5 \text{ m}$; allowable stress, $\sigma_a = 250 \text{ MPa}$; modulus of elasticity, $E = 210 \text{ GPa}$.
- Formulate the minimum mass design problem with constraints on member stresses and bounds on design variables. Solve the problem graphically using circular tubes as members.
- 3.27 Formulate and solve the problem of Exercise 2.1 graphically.
- 3.28 In the design of a closed-end, thin-walled cylindrical pressure vessel shown in Fig. E3.28, the design objective is to select the mean radius R and wall thickness t to minimize the total mass. The vessel should contain at least 25.0 m^3 of gas at an internal pressure of 3.5 MPa . It is required that the circumferential stress in the pressure vessel not exceed 210 MPa and the circumferential strain not exceed $(1.0\text{E} - 03)$. The circumferential stress and strain are calculated from the equations

$$\sigma_c = \frac{PR}{t}, \quad \epsilon_c = \frac{PR(2-\nu)}{2Et}$$

where ρ = mass density (7850 kg/m^3), σ_c = circumferential stress (Pa), ϵ_c = circumferential strain, P = internal pressure (Pa), E = Young's modulus (210 GPa), and ν = Poisson's ratio (0.3).

- (i) Formulate the optimum design problem and (ii) solve the problem graphically.

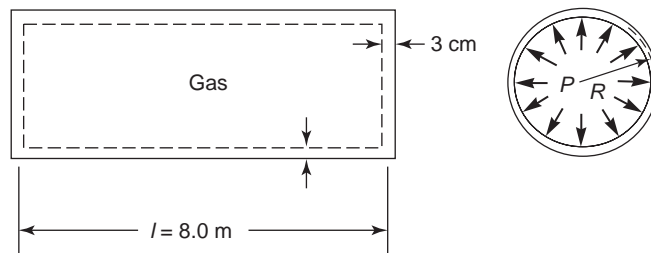


FIGURE E3-28 Cylindrical pressure vessel.

- 3.29 Consider the symmetric three-bar truss design problem formulated in Section 2.10. Formulate and solve the problem graphically for the following data: $l = 1.0 \text{ m}$; $P = 100 \text{ kN}$; $\theta = 30^\circ$; mass density, $\rho = 2800 \text{ kg/m}^3$; modulus of elasticity, $E = 70 \text{ GPa}$; allowable stress, $\sigma_a = 140 \text{ MPa}$; $\Delta_u = 0.5 \text{ cm}$; $\Delta_v = 0.5 \text{ cm}$; $\omega_o = 50 \text{ Hz}$; $\beta = 1.0$; $A_1, A_2 \geq 2 \text{ cm}^2$.

- 3.30 Consider the cabinet design problem given in Section 2.6. Use the equality constraints to eliminate three design variables from the problem. Restate the problem in terms of the remaining three variables, transcribing it into the standard form.
- 3.31 Solve the insulated spherical tank design problem formulated in Section 2.3 graphically for the following data: $r = 3.0$ m, $c_1 = \$100$, $c_2 = 500$, $c_3 = \$10$, $c_4 = \$5$, $\Delta T = 10$.
- 3.32 Solve the cylindrical tank design problem given in Section 2.8 graphically for the following data: $c = \$1500/\text{m}^2$, $V = 3000 \text{ m}^3$.
- 3.33 Consider the minimum mass tubular column problem formulated in Section 2.7. Find the optimum solution for the problem using the graphical method for the data: load, $P = 100$ kN; length, $l = 5.0$ m; Young's modulus, $E = 210$ GPa; allowable stress, $\sigma_a = 250$ MPa; mass density, $\rho = 7850 \text{ kg/m}^3$; $R \leq 0.4$ m; $t \leq 0.1$ m; $R, t \geq 0$.
- 3.34* Design a hollow torsion rod shown in Fig. E3.34 to satisfy the following requirements (created by J. M. Trummel):
1. The calculated shear stress, τ , shall not exceed the allowable shear stress τ_a under the normal operating torque T_o (N·m).
 2. The calculated angle of twist, θ , shall not exceed the allowable twist, θ_a (radians).
 3. The member shall not buckle under a short duration torque of T_{\max} (N·m).

Requirements for the rod and material properties are given in Tables E3.34(A) and E3.34(B) (select a material for one rod). Use the following design variables: $x_1 =$ outside diameter of the rod and $x_2 =$ ratio of inside/outside diameter, d_i/d_o .

Using graphical optimization, determine the inside and outside diameters for a minimum mass rod to meet the above design requirements. Compare the hollow rod with an equivalent solid rod ($d_i/d_o = 0$). Use consistent set of units (e.g., Newtons and millimeters) and let the minimum and maximum values for design variables be given as

$$0.02 \leq d_o \leq 0.5 \text{ m}, \quad 0.60 \leq \frac{d_i}{d_o} \leq 0.999$$

Useful expressions for the rod are:

$$\text{Mass of rod:} \quad M = \frac{\pi}{4} \rho l (d_o^2 - d_i^2), \quad \text{kg}$$

$$\text{Calculated shear stress:} \quad \tau = \frac{c}{J} T_o, \quad \text{Pa}$$

$$\text{Calculated angle of twist:} \quad \theta = \frac{l}{GJ} T_o, \quad \text{radians}$$

$$\text{Critical buckling torque:} \quad T_{cr} = \frac{\pi d_o^3 E}{12\sqrt{2}(1-\nu^2)^{0.75}} \left(1 - \frac{d_i}{d_o}\right)^{2.5}, \quad \text{N} \cdot \text{m}$$

Notation

$M =$ mass of the rod (kg),

$d_o =$ outside diameter of the rod (m),

$d_i =$ inside diameter of the rod (m),

$\rho =$ mass density of material (kg/m^3),

$l =$ length of the rod (m),

T_o = normal operating torque (N·m),
 c = distance from rod axis to extreme fiber (m),
 J = polar moment of inertia (m⁴),
 θ = angle of twist (radians),
 G = modulus of rigidity (Pa),
 T_{cr} = critical buckling torque (N·m),
 E = modulus of elasticity (Pa), and
 ν = Poisson's ratio.

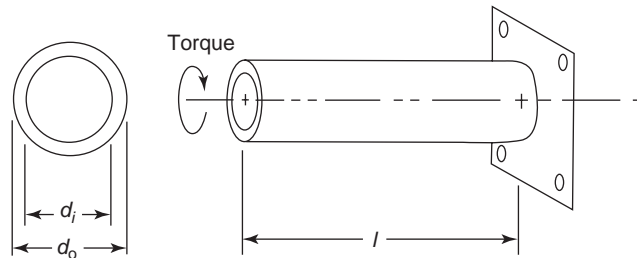


FIGURE E3-34 Hollow torsion rod.

TABLE E3-34(A) Rod Requirements

Torsion rod number	Length, l (m)	Normal torque, T_o (kN·m)	Maximum, T_{max} (kN·m)	Allowable twist, θ_a (degrees)
1	0.50	10.0	20.0	2
2	0.75	15.0	25.0	2
3	1.00	20.0	30.0	2

TABLE E3-34(B) Materials and Properties for the Torsion Rod

Material	Density, ρ (kg/m ³)	Allowable shear stress, τ_a (MPa)	Elastic modulus, E (GPa)	Shear modulus, G (GPa)	Poisson ratio (ν)
1. 4140 alloy steel	7850	275	210	80	0.30
2. Aluminum alloy 24 ST4	2750	165	75	28	0.32
3. Magnesium alloy A261	1800	90	45	16	0.35
4. Beryllium	1850	110	300	147	0.02
5. Titanium	4500	165	110	42	0.30

- 3.35* Formulate and solve Exercise 3.34 using the outside diameter d_o and the inside diameter d_i as design variables.
- 3.36* Formulate and solve Exercise 3.34 using the mean radius R and wall thickness t as design variables. Let the bounds on design variables be given as $5 \leq R \leq 20$ cm and $0.2 \leq t \leq 4$ cm.
- 3.37 Formulate the problem of Exercise 2.3 and solve it using the graphical method.
- 3.38 Formulate the problem of Exercise 2.4 and solve it using the graphical method.
- 3.39 Solve Exercise 3.23 for a column pinned at both ends. The buckling load for such a column is given as $\pi^2 EI/l^2$. Use graphical method.

- 3.40 Solve Exercise 3.23 for a column fixed at both ends. The buckling load for such a column is given as $4\pi^2 EI/l^2$. Use graphical method.
- 3.41 Solve Exercise 3.23 for a column fixed at one end and pinned at the other. The buckling load for such a column is given as $2\pi^2 EI/l^2$. Use graphical method.
- 3.42 Solve Exercise 3.24 for a column pinned at both ends. The buckling load for such a column is given as $\pi^2 EI/l^2$. Use graphical method.
- 3.43 Solve Exercise 3.24 for a column fixed at both ends. The buckling load for such a column is given as $4\pi^2 EI/l^2$. Use graphical method.
- 3.44 Solve Exercise 3.24 for a column fixed at one end and pinned at the other. The buckling load for such a column is given as $2\pi^2 EI/l^2$. Use graphical method.
- 3.45 Solve the can design problem formulated in Section 2.2 using the graphical approach.
- 3.46 Consider the two-bar truss shown in Fig. 2-2. Using the given data, design a minimum mass structure where $W = 100 \text{ kN}$; $\theta = 30^\circ$; $h = 1 \text{ m}$; $s = 1.5 \text{ m}$; modulus of elasticity, $E = 210 \text{ GPa}$; allowable stress, $\sigma_a = 250 \text{ MPa}$; mass density, $\rho = 7850 \text{ kg/m}^3$. Use Newtons and millimeters as units. The members should not fail in stress and their buckling should be avoided. Deflection at the top in either direction should not be more than 5 cm.
Use cross-sectional areas A_1 and A_2 of the two members as design variables and let the moment of inertia of the members be given as $I = A^2$. Areas must also satisfy the constraint $1 \leq A_i \leq 50 \text{ cm}^2$.
- 3.47 For Exercise 3.46, use hollow circular tubes as members with mean radius R and wall thickness t as design variables. Make sure that $R/t \leq 50$. Design the structure so that member 1 is symmetric with member 2. The radius and thickness must also satisfy the constraints $2 \leq t \leq 40 \text{ mm}$ and $2 \leq R \leq 40 \text{ cm}$.
- 3.48 Design a symmetric structure defined in Exercise 3.46 treating cross-sectional area A and height h as design variables. The design variables must also satisfy the constraints $1 \leq A \leq 50 \text{ cm}^2$ and $0.5 \leq h \leq 3 \text{ m}$.
- 3.49 Design a symmetric structure defined in Exercise 3.46 treating cross-sectional area A and the span s as design variables. The design variables must also satisfy the constraints $1 \leq A \leq 50 \text{ cm}^2$ and $0.5 \leq s \leq 4 \text{ m}$.
- 3.50 A minimum mass symmetric (area of member 1 is the same as member 3) three-bar truss is to be designed to support a load P as shown in Fig. 2-6. The following notation may be used: $P_u = P \cos\theta$, $P_v = P \sin\theta$, $A_1 =$ cross-sectional area of members 1 and 3, $A_2 =$ cross-sectional area of member 2.
The members must not fail under the stress, and deflection at node 4 must not exceed 2 cm in either direction. Use Newtons and millimeters as units. The data is given as $P = 50 \text{ kN}$; $\theta = 30^\circ$; mass density, $\rho = 7850 \text{ kg/m}^3$; modulus of elasticity, $E = 210 \text{ GPa}$; allowable stress, $\sigma_a = 150 \text{ MPa}$. The design variables must also satisfy the constraints $50 \leq A_i \leq 5000 \text{ mm}^2$.
- 3.51* **Design of a water tower support column.** As a member of the ABC Consulting Engineers you have been asked to design a cantilever cylindrical support column of minimum mass for a new water tank. The tank itself has already been designed in the tear-drop shape shown in Fig. E3-51. The height of the base of the tank (H), the diameter of the tank (D), and the wind pressure on the tank (w) are given as $H = 30 \text{ m}$, $D = 10 \text{ m}$, and $w = 700 \text{ N/m}^2$. Formulate the design optimization problem and solve it graphically (created by G. Baenziger).

In addition to designing for combined axial and bending stresses and buckling, several limitations have been placed on the design. The support column must have an inside diameter of at least 0.70 m (d_i) to allow for piping and ladder access to the interior of the tank. To prevent local buckling of the column walls the diameter/thickness ratio (d_o/t) shall not be greater than 92. The large mass of water and steel makes deflections critical as they add to the bending moment. The deflection effects as well as an assumed construction eccentricity (e) of 10 cm must be accounted for in the design process. Deflection at C.G. of the tank should not be greater than Δ .

Limits on the inner radius and wall thickness are $0.35 \leq R \leq 2.0$ m and $1.0 \leq t \leq 20$ cm.

Pertinent constants and formulas

Height of water tank,	$h = 10$ m
Allowable deflection,	$\Delta = 20$ cm
Unit weight of water,	$\gamma_w = 10$ kN/m ³
Unit weight of steel,	$\gamma_s = 80$ kN/m ³
Modulus of elasticity,	$E = 210$ GPa
Moment of inertia of the column,	$I = \frac{\pi}{64} [d_o^4 - (d_o - 2t)^4]$
Cross-sectional area of column material,	$A = \pi t(d_o - t)$
Allowable bending stress,	$\sigma_b = 165$ MPa
Allowable axial stress,	$\sigma_a = \frac{12\pi^2 E}{92(H/r)^2}$ (calculated using the critical buckling load with a factor of safety of $\frac{23}{12}$)
Radius of gyration,	$r = \sqrt{I/A}$
Average thickness of tank wall,	$t_t = 1.5$ cm
Volume of tank,	$V = 1.2\pi D^2 h$
Surface area of tank,	$A_s = 1.25\pi D^2$
Projected area of tank, for wind loading,	$A_p = \frac{2Dh}{3}$
Load on the column due to weight of water and steel tank,	$P = V\gamma_w + A_s t_t \gamma_s$
Lateral load at the tank C.G. due to wind pressure, $W = wA_p$.	
Deflection at C.G. of tank, $\delta = \delta_1 + \delta_2$, where	
	$\delta_1 = \frac{WH^2}{12EI} (4H + 3h)$
	$\delta_2 = \frac{H}{2EI} (0.5Wh + Pe)(H + h)$
Moment at base,	$M = W(H + 0.5h) + (\delta + e)P$
Bending stress,	$f_b = \frac{M}{2I} d_o$
Axial stress,	$f_a (= P/A) = \frac{V\gamma_w + A_s \gamma_s t_t}{\pi t(d_o - t)}$
Combined stress constraint,	$\frac{f_a}{\sigma_a} + \frac{f_b}{\sigma_b} \leq 1$
Gravitational acceleration,	$g = 9.81$ m/s ²

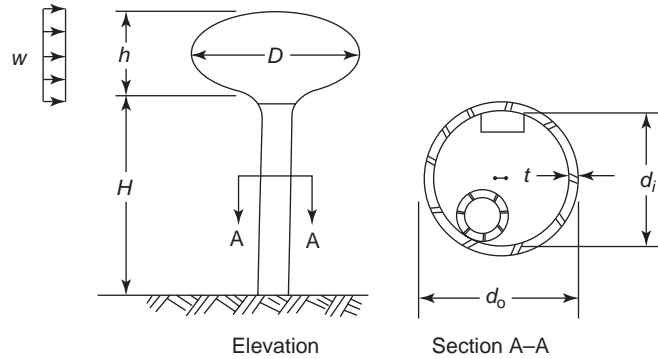


FIGURE E3-51 Water tower support column.

3.52* **Design of a flag pole.** Your consulting firm has been asked to design a minimum mass flag pole of height H . The pole will be made of uniform hollow circular tubing with d_o and d_i as outer and inner diameters, respectively. The pole must not fail under the action of high winds.

For design purposes, the pole will be treated as a cantilever that is subjected to a uniform lateral wind load of w (kN/m). In addition to the uniform load, the wind induces a concentrated load of P (kN) at the top of the pole, as shown in Fig. E3.52. The flag pole must not fail in bending or shear. The deflection at the top should not exceed 10 cm. The ratio of mean diameter to thickness must not exceed 60. The pertinent data are given below. Assume any other data if needed. The minimum and maximum values of design variables are $5 \leq d_o \leq 50$ cm and $4 \leq d_i \leq 45$ cm.

Pertinent constants and equations

Cross-sectional area,	$A = \frac{\pi}{4}(d_o^2 - d_i^2)$
Moment of inertia,	$I = \frac{\pi}{64}(d_o^4 - d_i^4)$
Modulus of elasticity,	$E = 210$ GPa
Allowable bending stress,	$\sigma_b = 165$ MPa
Allowable shear stress,	$\tau_s = 50$ MPa
Mass density,	$\rho = 7800$ kg/m ³
Wind load,	$w = 2.0$ kN/m
Height of flag pole,	$H = 10$ m
Concentrated load at top,	$P = 4.0$ kN
Moment at the base,	$M = (PH + 0.5wH^2)$, kN·m
Bending stress,	$\sigma = \frac{M}{2I}d_o$, kPa
Shear at the base,	$S = (P + wH)$, kN
Shear stress,	$\tau = \frac{S}{12I}(d_o^2 + d_o d_i + d_i^2)$, kPa
Deflection at the top,	$\delta = \frac{PH^3}{3EI} + \frac{wH^4}{8EI}$
Minimum and maximum thickness,	0.5 and 2 cm

Formulate the design problem and solve it using the graphical optimization technique.

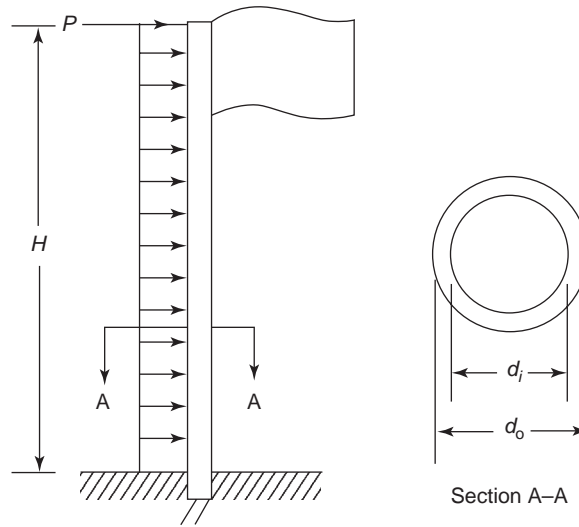


FIGURE E3-52 Flag pole.

3.53* **Design of a sign support column.** The design department of a company has been asked to design a support column of minimum weight for the sign shown. The height to the bottom of the sign H , the width of the sign b , and the wind pressure p on the sign are as follows: $H = 20$ m, $b = 8$ m, $p = 800$ N/m² (Fig. E3.53).

The sign itself weighs 2.5 kN/m²(w). The column must be safe with respect to combined axial and bending stresses. The allowable axial stress includes a factor of safety with respect to buckling. To prevent local buckling of the plate the diameter/thickness ratio d_o/t must not exceed 92. Note that the bending stress in the column will increase as a result of the deflection of the sign under the wind load. The maximum deflection at the center of gravity of the sign should not exceed 0.1 m. The minimum and maximum values of design variables are $25 \leq d_o \leq 150$ cm and $0.5 \leq t \leq 10$ cm (created by H. Kane).

Pertinent constants and equations

Height of the sign, $h = 4.0$ m

For column section

Area, $A = \frac{\pi}{4} [d_o^2 - (d_o - 2t)^2]$

Moment of inertia, $I = \frac{\pi}{64} (d_o^4 - (d_o - 2t)^4)$

Radius of gyration, $r = \sqrt{I/A}$

Young's modulus (aluminum alloy), $E = 75$ GPa

Unit weight of steel, $\gamma = 80$ kN/m³

Allowable bending stress, $\sigma_b = 140$ MPa

Allowable axial stress,	$\sigma_a = \frac{12\pi^2 E}{92(H/r)^2}$
Wind force,	$F = pbh$
Weight of sign,	$W = wbh$
Deflection at center of gravity of sign,	$\delta = \frac{F}{EI} \left(\frac{H^3}{3} + \frac{H^2 h}{2} + \frac{Hh^2}{4} \right)$
Bending stress in column,	$f_b = \frac{M}{2I} d_o$
Axial stress,	$f_a = \frac{W}{A}$
Moment at the base,	$M = F \left(H + \frac{h}{2} \right) + W\delta$
Combined stress requirement,	$\frac{f_a}{\sigma_a} + \frac{f_b}{\sigma_b} \leq 1$

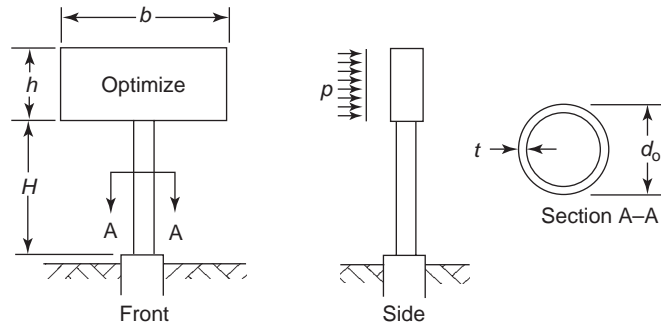


FIGURE E3-53 Sign support column.

3.54* **Design of a tripod.** Design a minimum mass tripod of height H to support a vertical load $W = 60$ kN. The tripod base is an equilateral triangle with sides $B = 1200$ mm. The struts have a solid circular cross section of diameter D (Fig. E3-54).

The axial stress in the struts must not exceed the allowable stress in compression, and the axial load in the strut P must not exceed the critical buckling load P_{cr} divided by a safety factor $FS = 2$. Use consistent units of Newtons and centimeters. The minimum and maximum values for design variables are $0.5 \leq H \leq 5$ m and $0.5 \leq D \leq 50$ cm. Material properties and other relationships are given below:

Material: aluminum alloy 2014-T6

Allowable compressive stress,

$$\sigma_a = 150 \text{ MPa}$$

Young's modulus,

$$E = 75 \text{ GPa}$$

Mass density,

$$\rho = 2800 \text{ kg/m}^3$$

Strut length,

$$l = \left(H^2 + \frac{1}{3} B^2 \right)^{0.5}$$

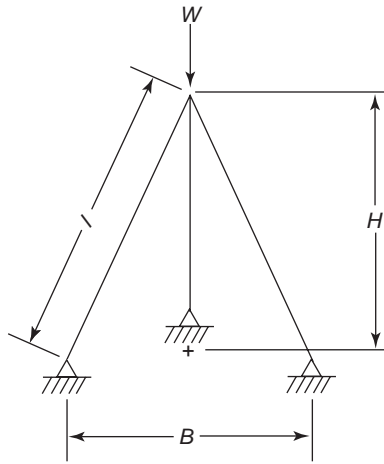


FIGURE E3-54 A tripod.

Critical buckling load,	$P_{cr} = \frac{\pi^2 EI}{l^2}$
Moment of inertia,	$I = \frac{\pi}{64} D^4$
Strut load,	$P = \frac{Wl}{3H}$

4 Optimum Design Concepts

Upon completion of this chapter, you will be able to:

- Define local and global minima (maxima) for unconstrained and constrained problems
- Write optimality conditions for unconstrained and constrained problems
- Check optimality of a given point for unconstrained and constrained problems
- Solve first-order optimality conditions for candidate minimum points
- Check convexity of a function and the design optimization problem
- Use Lagrange multipliers to study changes to the optimum value of the cost function due to constraint variations

In this chapter, we discuss *basic ideas, concepts, and theories used for design optimization* (the minimization problem). Theorems on the subject are stated without proofs. Their implications and use in the optimization process are discussed. The student is reminded to review the basic terminology and notation explained in Section 1.5 as they are used throughout the present chapter and the remaining text.

As an overview of the material of the present and the remaining chapters, we show in Fig. 4-1 a broad classification of the optimization techniques. Two philosophically different viewpoints are shown. It is important to understand the features—limitations and advantages—of the two approaches to gain insights for practical applications of optimization. The two categories are indirect (or optimality criteria) methods and direct (or search) methods. *Optimality criteria* are the conditions a function must satisfy at its minimum point. Minimization techniques seeking solutions to optimality conditions are often called *indirect methods*. The *direct (search) techniques* are based on a different philosophy. There we start with an estimate of the optimum design for the problem. Usually the starting design will not satisfy optimality criteria; therefore, it is improved iteratively until they are satisfied. Thus, in this approach we search the design space for optimum points. We shall address unconstrained and constrained optimization problems under both categories in this text.

A thorough knowledge of *optimality conditions* is required to understand the performance of various *numerical (search) methods* discussed later in the text. This chapter focuses on the discussion of the optimality conditions and the solution methods based on them. Simple

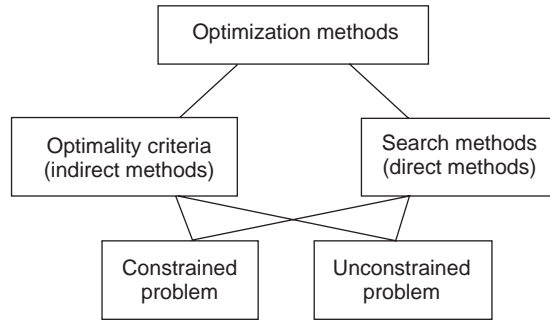


FIGURE 4-1 Classification of optimization methods.

examples are used to explain the underlying concepts and ideas. The examples will also show practical limitations of the methods based on optimality conditions. The *search methods* are presented in later chapters and refer to the results discussed in this chapter. Therefore, the *material in the present chapter should be understood thoroughly*. We will first discuss the concept of local optimum of a function and the conditions that characterize it. The problem of global optimality of a function will be discussed later in this chapter. *It is important to note that all the problem functions are assumed twice continuously differentiable*.

4.1 Definitions of Global and Local Minima

Optimality conditions for a minimum point of the function are discussed in later sections. In this section, concepts of *local and global minima* are defined and illustrated using the *standard mathematical model for design optimization* defined in Chapter 2. The design optimization problem is always converted to minimization of a cost function subject to equality and inequality constraints. The problem is re-stated as follows: Find design variable vector \mathbf{x} to minimize a *cost function* $f(\mathbf{x})$ subject to the *equality constraints* $h_j(\mathbf{x}) = 0$, $j = 1$ to p and *inequality constraints* $g_i(\mathbf{x}) \leq 0$, $i = 1$ to m . Note that the simple bounds on design variables, such as $x_i \geq 0$, or $x_{il} \leq x_i \leq x_{iu}$ are assumed to be included in the standard inequality constraints $g_i(\mathbf{x})$; x_{il} and x_{iu} are the smallest and largest allowed values for x_i . This is done to explain optimization concepts without getting bogged down with a separate treatment of these constraints. However, in numerical methods, these constraints are treated explicitly to take advantage of their special form.

4.1.1 Minimum

In Section 2.11, we defined the *feasible set* S (also called *constraint set*, *feasible region* or *feasible design space*) for a design problem as a collection of feasible designs:

$$S = \{\mathbf{x} | h_j(\mathbf{x}) = 0, \quad j = 1 \text{ to } p; \quad g_i(\mathbf{x}) \leq 0; \quad i = 1 \text{ to } m\}$$

Since there are no constraints in unconstrained problems, the entire design space is feasible for them. The *optimization problem* is to find a point in the feasible design space that gives a minimum value to the cost function. Methods to locate optimum designs are discussed throughout the text. We must first carefully define what is meant by an optimum. In the following discussion, \mathbf{x}^* is used to designate a particular point of the feasible set.

Global (Absolute) Minimum A function $f(\mathbf{x})$ of n variables has global (absolute) minimum at \mathbf{x}^* if

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad (4.1)$$

for all \mathbf{x} in the feasible design space S . If strict inequality holds for all \mathbf{x} other than \mathbf{x}^* in Eq. (4.1), then \mathbf{x}^* is called a *strong (strict) global minimum*; otherwise it is called a *weak global minimum*.

Local (Relative) Minimum A function $f(\mathbf{x})$ of n variables has a local (relative) minimum at \mathbf{x}^* if Inequality (4.1) holds for all \mathbf{x} in a small *neighborhood* N of \mathbf{x}^* in the feasible design space S . 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 the point \mathbf{x}^* is defined as the set of points

$$N = \{\mathbf{x} | \mathbf{x} \in S \text{ with } \|\mathbf{x} - \mathbf{x}^*\| < \delta\}$$

for some small $\delta > 0$. Geometrically, it is a small feasible region around the point \mathbf{x}^* . Note that a function $f(\mathbf{x})$ can have *strict global minimum* at only one point. It may, however, have a global minimum at several points if it has the same value at each of those points. Similarly, a function $f(\mathbf{x})$ can have a *strict local minimum* at only one point in the neighborhood N of \mathbf{x}^* . It may, however, have local minimum at several points in N if the function value is the same at each of those points. Note that *global* and *local maxima* are defined in a similar manner by simply reversing the inequality in Eq. (4.1). We also note here that these definitions do not provide a method for locating minimum points. Based on them, however, we can develop analyses and computational procedures to locate them. Also, we can use the definitions to check optimality of points in the graphical solution process presented in Chapter 3.

To understand the *graphical significance* of global and local minima, consider graphs of a function $f(x)$ of one variable, as shown in Fig. 4-2. In Part (A) of the figure, where x is between $-\infty$ and ∞ ($-\infty \leq x \leq \infty$), points B and D are local minima since the function has its smallest value in their neighborhood. Similarly, both A and C are points of local maxima for the function. There is, however, no global minimum or maximum for the function since the domain and the function $f(x)$ are unbounded, i.e., x and $f(x)$ are allowed to have any value between $-\infty$ and ∞ . If we restrict x to lie between $-a$ and b as in Part (B) of Fig. 4-2, then point E gives the global minimum and F the global maximum for the function. We shall further illustrate these concepts for constrained problems with Examples 4.1 to 4.3.

EXAMPLE 4.1 Graphical Representation of Unconstrained Minimum for a Constrained Problem

An optimum design problem is formulated and transcribed into the standard form in terms of the variables x and y as follows: minimize $f(x,y) = (x - 4)^2 + (y - 6)^2$ subject to the constraints:

$$g_1 = x + y - 12 \leq 0$$

$$g_2 = x - 8 \leq 0$$

$$g_3 = -x \leq 0 \quad (x \geq 0)$$

$$g_4 = -y \leq 0 \quad (y \geq 0)$$

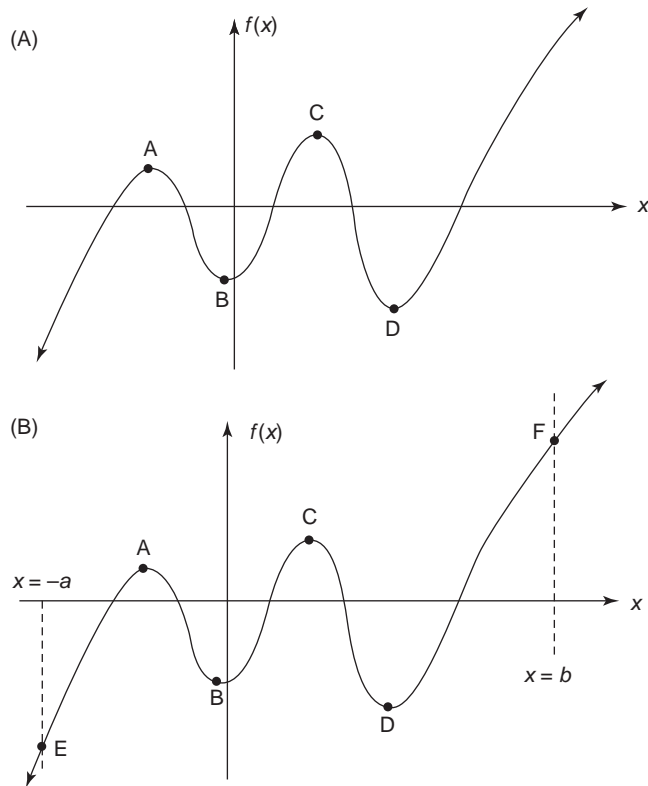


FIGURE 4-2 Graphical representation of optimum points. (A) Unbounded domain and function (no global optimum). (B) Bounded domain and function (global minimum and maximum exist).

Find local and global minima for the function $f(x,y)$ using the graphical method.

Solution. Using the procedure for graphical optimization described in Chapter 3, the constraints for the problem are plotted and the feasible region is identified as ABCD in Fig. 4-3. Contours of the cost function $f(x,y)$, which is an equation of a circle with center at (4, 6), are also shown. To locate the minimum points, we use the definition of local minimum and check the inequality $f(x^*,y^*) \leq f(x,y)$ at a candidate feasible point (x^*,y^*) in its small feasible neighborhood. Note that the cost function always has a nonnegative value at any point with the smallest value as zero at its center. Since the center of the circle at E(4, 6) is feasible, it is a local minimum point. We check the local minimum condition at some other points as follows:

Point A(0,0): $f(0,0) = 52$ is not a minimum point because the inequality $f(0,0) \leq f(x,y)$ is violated for any small feasible move away from the point A; i.e., the cost function reduces as we move away from the point A in the feasible region.

Point F(4,0): $f(4,0) = 36$ is also not a minimum point since there are feasible moves from the point for which the cost function can be reduced.

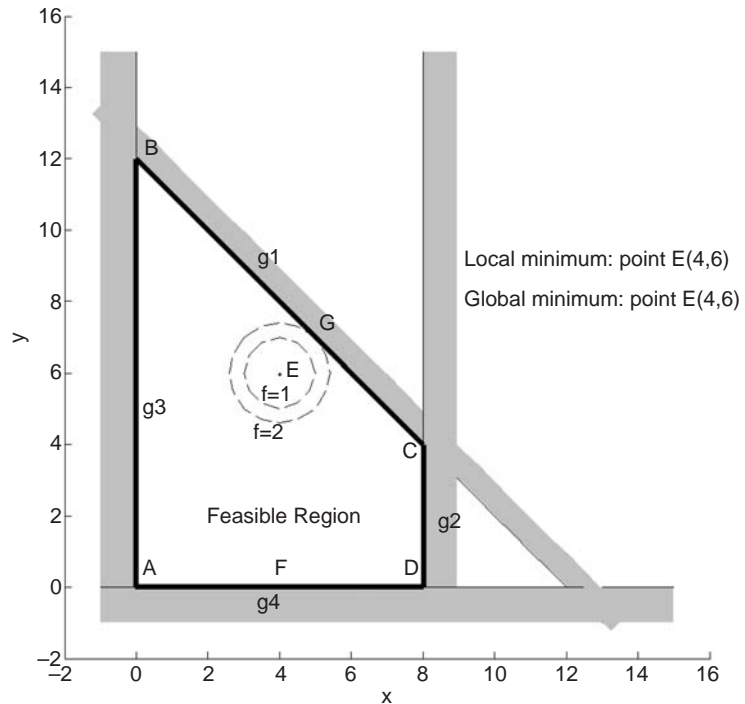


FIGURE 4-3 Graphical representation of unconstrained minimum for Example 4.1.

It can be checked that points B, C, D, and G are also not local minimum points. In fact, there is no other local minimum point. Thus, point E is a local as well as a global minimum point for the function. It is important to note that at the minimum point no constraints are active; i.e., *constraints play no role in determining the minimum points for this problem*. However, this is not always true, as we shall see in Example 4.2.

EXAMPLE 4.2 Graphical Representation of Constrained Minimum

Solve the optimum design problem formulated in terms of variables x and y as: minimize $f(x,y) = (x - 10)^2 + (y - 8)^2$ subject to the constraints of Example 4.1.

Solution. The feasible region for the problem is ABCD as shown in Fig. 4-4. The cost function is an equation of a circle with center at the point E(10, 8). However, the point (10, 8) is infeasible. Some cost contours are shown in the figure. The problem now is to find a point of the feasible region that is closest to the point E; i.e., with the smallest value for the cost function. It is seen that point G with coordinates (7, 5) and $f = 18$ has the smallest distance from point E. At this point, the constraint g_1 is active. Thus, *for the present objective function, the constraints play a prominent role in determining the minimum point for the problem*.

Use of the definition of a local minimum point also indicates that the point G is indeed a local minimum for the function since any feasible move from G results in an increase in the cost function. The use of the definition also indicates that there is no other local minimum point. Thus, point G is a global minimum point as well.

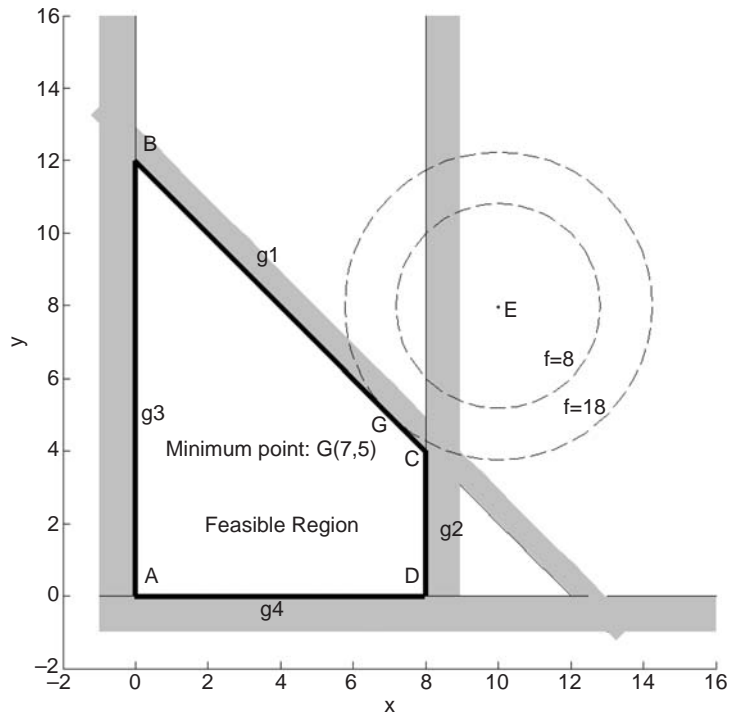


FIGURE 4-4 Graphical representation of constrained minimum for Example 4.2.

EXAMPLE 4.3 Graphical Representation of Maxima

Solve the optimum design problem formulated in terms of variables x and y as: maximize $f(x,y) = (x - 4)^2 + (y - 6)^2$ subject to the constraints of Example 4.1.

Solution. The feasible region for the problem is ABCD as shown in Fig. 4-3. The objective function is an equation of a circle with center at the point E(4, 6). Some objective function contours are shown in the figure. It is seen that point D(8, 0) is a local maximum point because any feasible move away from the point results in reduction of the objective function. Point C(8, 4) is not a local maximum point since a feasible move along the line CD results in an increase in the objective function thus violating the definition of a local max point [$f(x^*,y^*) \geq f(x,y)$]. It can be verified that points A and B are also local maximum points, and point G is not. Thus this problem has the following three local maximum points:

$$\text{Point A (0, 0): } f(0, 0) = 52$$

$$\text{Point B (0, 12): } f(0, 12) = 52$$

$$\text{Point D (8, 0): } f(8, 0) = 52$$

It is seen that the objective function has the same value at all the three points. Therefore all the points are global maximum points. This example shows that *an objective function can have several global optimum points in the feasible region.*

4.1.2 Existence of Minimum

In general we do not know before attempting to solve a problem if a minimum even exists. In certain cases we can *ensure existence of a minimum* even though we may not know how to find it. The Weierstrass theorem guarantees this when certain conditions are satisfied.

Theorem 4.1 Weierstrass Theorem—Existence of Global Minimum If $f(\mathbf{x})$ is continuous on a nonempty feasible set S that is closed and bounded, then $f(\mathbf{x})$ has a global minimum in S .

To use the theorem we must understand the meaning of a *closed and bounded set*. A set S is *closed* if it includes all its boundary points and every sequence of points has a subsequence that converges to a point in the set. A set is *bounded* if for any point, $\mathbf{x} \in S$, $\mathbf{x}^T \mathbf{x} < c$, where c is a finite number. Since the domain of the function in Fig. 4-2(A) is not closed and the function is also unbounded, a global minimum or maximum for the function is not assured. Actually, there is no global minimum or maximum for the function. However, in Fig. 4-2(B), since the feasible region is closed and bounded with $-a \leq x \leq b$ and the function is continuous, it has global minimum as well as maximum points. It is important to note that in general it is difficult to check the boundedness condition $\mathbf{x}^T \mathbf{x} < c$ since there are infinite points in S . The foregoing examples are simple where a graphical representation of the problem is available and it is easy to check the condition. Nevertheless it is important to keep the theorem in mind while using a numerical method to solve an optimization problem. If the numerical process is not converging to a solution, then perhaps some conditions of this theorem are violated and the problem formulation needs to be re-examined carefully. Example 4.4 further illustrates the use of Weierstrass theorem.

EXAMPLE 4.4 Existence of Global Minimum Using Weierstrass Theorem

Consider a function $f(x) = -1/x$ defined on the set $S = \{x \mid 0 < x \leq 1\}$. Check existence of a global minimum for the function.

Solution. The feasible set S is not closed since it does not include the boundary point $x = 0$. The conditions of the Weierstrass theorem are not satisfied, although f is continuous on S . Existence of a global minimum is not guaranteed and indeed there is no point x^* satisfying $f(x^*) \leq f(x)$ for all $x \in S$. If we define $S = \{x \mid 0 \leq x \leq 1\}$, then the feasible set is closed and bounded. However, f is not defined at $x = 0$ (hence not continuous), so the conditions of the theorem are still not satisfied and there is no guarantee of a global minimum for f in the set S .

Note that when conditions of the Weierstrass theorem are satisfied, existence of a global optimum is guaranteed. It is important, however, to realize that when they are not satisfied, a global solution may still exist. The theorem does not rule out this possibility. The difference is that we cannot guarantee its existence. Note also that the theorem does not give a method for finding a global solution even if its conditions are satisfied; it is only an existence theorem.

4.2 Review of Some Basic Calculus Concepts

Optimality conditions for a minimum point are discussed in later sections. Since most optimization problems involve functions of several variables, these conditions use ideas from vector calculus. Therefore, in this section, *we review basic concepts from calculus using the*

vector and matrix notations. Basic material related to vector and matrix algebra (linear algebra) is described in Appendix B. It is important to be comfortable with these materials in order to understand the optimality conditions. The topics from this material may be covered as a review all at once or they may be reviewed on an “as needed” basis at an appropriate time during coverage of various topics from this chapter.

The *differentiation notation* for functions of several variables is introduced. The *gradient vector* for a function of several variables requiring first partial derivatives of the function is defined. The *Hessian matrix* for the function requiring second partial derivatives of the function is then defined. *Taylor’s expansions* for functions of single and multiple variables are discussed. The idea of Taylor series is fundamental to the development of optimum design concepts and numerical methods, so it should be thoroughly understood. The concept of *quadratic forms* is needed to discuss sufficiency conditions for optimality. Therefore, notation and analyses related to quadratic forms are described. The concepts of necessary and sufficient conditions are explained.

4.2.1 Gradient Vector

Since the gradient of a function is used while discussing methods of optimum design, we define and discuss its geometrical significance. Also, the *differentiation notation* defined here is used throughout the text. Therefore, it should be clearly understood.

Consider a function $f(\mathbf{x})$ of n variables x_1, x_2, \dots, x_n . The partial derivative of the function with respect to x_1 at a given point \mathbf{x}^* is defined as $\partial f(\mathbf{x}^*)/\partial x_1$, with respect to x_2 as $\partial f(\mathbf{x}^*)/\partial x_2$, and so on. Let c_i represent the partial derivative of $f(\mathbf{x})$ with respect to x_i at the point \mathbf{x}^* . Then using the index notation of Section 1.5, we can represent all partial derivatives of $f(\mathbf{x})$ as follows:

$$c_i = \frac{\partial f(\mathbf{x}^*)}{\partial x_i}; \quad i = 1 \text{ to } n \quad (4.2)$$

For convenience and compactness of notation, we arrange the partial derivatives $\partial f(\mathbf{x}^*)/\partial x_1, \partial f(\mathbf{x}^*)/\partial x_2, \dots, \partial f(\mathbf{x}^*)/\partial x_n$ into a column vector called the *gradient vector* and represent it by any of the following symbols: $\mathbf{c}, \nabla f, \partial f/\partial \mathbf{x}, \text{grad } f$, as

$$\mathbf{c} = \nabla f(\mathbf{x}^*) = \begin{bmatrix} \frac{\partial f(\mathbf{x}^*)}{\partial x_1} \\ \frac{\partial f(\mathbf{x}^*)}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x}^*)}{\partial x_n} \end{bmatrix} = \left[\frac{\partial f(\mathbf{x}^*)}{\partial x_1} \quad \frac{\partial f(\mathbf{x}^*)}{\partial x_2} \quad \dots \quad \frac{\partial f(\mathbf{x}^*)}{\partial x_n} \right]^T \quad (4.3)$$

where superscript T denotes transpose of a vector or a matrix. Note that all partial derivatives are calculated at the given point \mathbf{x}^* . That is, each component of the gradient vector is a function in itself which must be evaluated at the given point \mathbf{x}^* .

*Geometrically, the gradient vector is normal to the tangent plane at the point \mathbf{x}^** as shown in Fig. 4-5 for a function of three variables. Also, it points in the direction of *maximum increase* in the function. These properties are quite important, and will be proved and discussed in Chapter 9. They will be used in developing optimality conditions and numerical methods for optimum design. In Example 4.5 the gradient vector for a function is calculated.

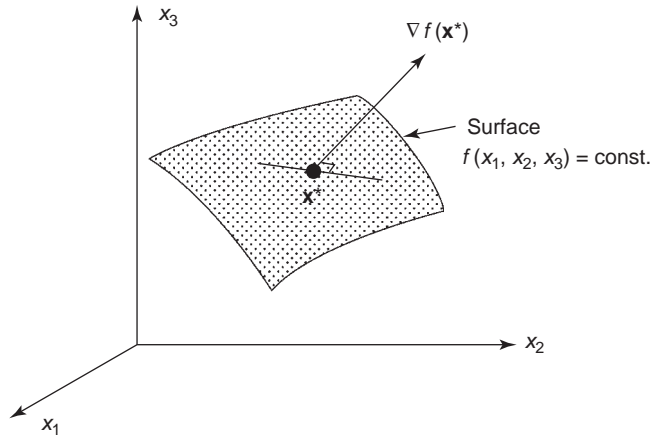


FIGURE 4-5 Gradient vector for $f(x_1, x_2, x_3)$ at the point \mathbf{x}^* .

EXAMPLE 4.5 Calculation of Gradient Vector

Calculate the gradient vector for the function $f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 - 1)^2$ at the point $\mathbf{x}^* = (1.8, 1.6)$.

Solution. The given function is the equation for a circle with center at the point $(1, 1)$. Since $f(1.8, 1.6) = (1.8 - 1)^2 + (1.6 - 1)^2 = 1$, the point $(1.8, 1.6)$ lies on a circle of radius 1, shown as point A in Fig. 4-6. The partial derivatives for the function at point $(1.8, 1.6)$ are calculated as

$$\frac{\partial f}{\partial x_1}(1.8, 1.6) = 2(x_1 - 1) = 2(1.8 - 1) = 1.6$$

$$\frac{\partial f}{\partial x_2}(1.8, 1.6) = 2(x_2 - 1) = 2(1.6 - 1) = 1.2$$

Thus, the gradient vector for $f(\mathbf{x})$ at point $(1.8, 1.6)$ is given as $\mathbf{c} = (1.6, 1.2)$. This is shown in Fig. 4-6. It can be seen that vector \mathbf{c} is normal to the circle at point $(1.8, 1.6)$. This is consistent with the observation that gradient is normal to the surface.

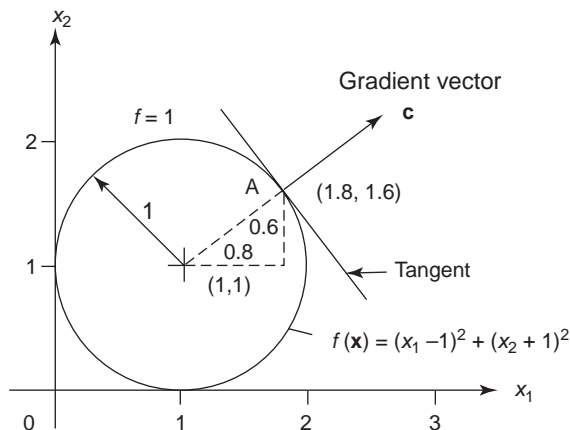


FIGURE 4-6 Gradient vector for the function $f(\mathbf{x})$ of Example 4-5 at the point $(1.8, 1.6)$.

4.2.2 Hessian Matrix

Differentiating the gradient vector once again, we obtain a matrix of second partial derivatives for the function $f(\mathbf{x})$ called the *Hessian matrix*, or simply the Hessian. That is, differentiating each component of the gradient vector given in Eq. (4.3) with respect to x_1, x_2, \dots, x_n , we obtain

$$\frac{\partial^2 f}{\partial \mathbf{x} \partial \mathbf{x}} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (4.4)$$

where all derivatives are calculated at the given point \mathbf{x}^* . The Hessian is an $n \times n$ matrix, also denoted as \mathbf{H} or $\nabla^2 f$. *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, since $f(\mathbf{x})$ is assumed to be twice continuously differentiable, the cross partial derivatives are equal, i.e.,

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}; \quad i = 1 \text{ to } n, j = 1 \text{ to } n$$

Therefore, *the Hessian is always a symmetric matrix.* It plays a prominent role in the sufficiency conditions for optimality as discussed later in this chapter. It will be written as

$$H = \left[\frac{\partial^2 f}{\partial x_j \partial x_i} \right]; \quad i = 1 \text{ to } n, j = 1 \text{ to } n \quad (4.5)$$

The gradient and Hessian of a function are calculated in Example 4.6.

EXAMPLE 4.6 Evaluation of Gradient and Hessian of a Function

For the following function, calculate the gradient vector and the Hessian matrix at the point (1, 2):

$$f(\mathbf{x}) = x_1^3 + x_2^3 + 2x_1^2 + 3x_2^2 - x_1x_2 + 2x_1 + 4x_2 \quad (a)$$

Solution. The first partial derivatives of the function are given as

$$\frac{\partial f}{\partial x_1} = 3x_1^2 + 4x_1 - x_2 + 2; \quad \frac{\partial f}{\partial x_2} = 3x_2^2 + 6x_2 - x_1 + 4 \quad (b)$$

Substituting the point $x_1 = 1, x_2 = 2$, the gradient vector is given as: $\mathbf{c} = (7, 27)$. The second partial derivatives of the function are calculated as

$$\frac{\partial^2 f}{\partial x_1^2} = 6x_1 + 4; \quad \frac{\partial^2 f}{\partial x_1 \partial x_2} = -1; \quad \frac{\partial^2 f}{\partial x_2 \partial x_1} = -1; \quad \frac{\partial^2 f}{\partial x_2^2} = 6x_2 + 6. \quad (c)$$

Therefore, the Hessian matrix at the point (1, 2) is given as

$$\mathbf{H}(1, 2) = \begin{bmatrix} 10 & -1 \\ -1 & 18 \end{bmatrix} \quad (d)$$

4.2.3 Taylor's Expansion

A function can be approximated by polynomials in a neighborhood of any point in terms of its value and derivatives using Taylor's expansion. Consider first a function $f(x)$ of a single variable. Taylor's expansion for $f(x)$ about the point x^* is

$$f(x) = f(x^*) + \frac{df(x^*)}{dx}(x - x^*) + \frac{1}{2} \frac{d^2 f(x^*)}{dx^2}(x - x^*)^2 + R \quad (4.6)$$

where R is the *remainder* term that is smaller in magnitude than the previous terms if x is sufficiently close to x^* . If we let $x - x^* = d$ (a small change in the point x^*), Taylor's expansion of Eq. (4.6) becomes

$$f(x^* + d) = f(x^*) + \frac{df(x^*)}{dx}d + \frac{1}{2} \frac{d^2 f(x^*)}{dx^2}d^2 + R \quad (4.7)$$

For a function of two variables $f(x_1, x_2)$, Taylor's expansion at the point (x_1^*, x_2^*) is

$$\begin{aligned} f(x_1, x_2) = & f(x_1^*, x_2^*) + \frac{\partial f}{\partial x_1}(x_1 - x_1^*) + \frac{\partial f}{\partial x_2}(x_2 - x_2^*) \\ & + \frac{1}{2} \left[\frac{\partial^2 f}{\partial x_1^2}(x_1 - x_1^*)^2 + 2 \frac{\partial^2 f}{\partial x_1 \partial x_2}(x_1 - x_1^*)(x_2 - x_2^*) + \frac{\partial^2 f}{\partial x_2^2}(x_2 - x_2^*)^2 \right] + R \end{aligned} \quad (4.8)$$

where all partial derivatives are calculated at the given point (x_1^*, x_2^*) . For notational compactness, the arguments of these partial derivatives are omitted in Eq. (4.8) and in all subsequent discussions. Taylor's expansion in Eq. (4.8) can be written using the *summation notation* defined in Section 1.5 as

$$\begin{aligned} f(x_1, x_2) = & f(x_1^*, x_2^*) + \sum_{i=1}^2 \frac{\partial f}{\partial x_i}(x_i - x_i^*) \\ & + \frac{1}{2} \sum_{i=1}^2 \sum_{j=1}^2 \frac{\partial^2 f}{\partial x_i \partial x_j}(x_i - x_i^*)(x_j - x_j^*) + R \end{aligned} \quad (4.9)$$

It can be seen that by expanding the summations in Eq. (4.9), Eq. (4.8) is obtained. Recognizing the quantities $\partial f / \partial x_i$ as components of the gradient of the function given in Eq. (4.3) and $\partial^2 f / \partial x_i \partial x_j$ as the Hessian of Eq. (4.5) evaluated at the given point \mathbf{x}^* , Taylor's expansion can also be written in *matrix notation* as

$$f(\mathbf{x}) = f(\mathbf{x}^*) + \nabla f^T(\mathbf{x} - \mathbf{x}^*) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T \mathbf{H}(\mathbf{x} - \mathbf{x}^*) + R \quad (4.10)$$

where $\mathbf{x} = (x_1, x_2)$, $\mathbf{x}^* = (x_1^*, x_2^*)$, and \mathbf{H} is the 2×2 Hessian matrix. Note that with matrix notation, Taylor's expansion in Eq. (4.10) can be easily generalized to functions of n variables. In that case, \mathbf{x} , \mathbf{x}^* , and ∇f are n dimensional vectors and \mathbf{H} is the $n \times n$ Hessian matrix. Defining $\mathbf{x} - \mathbf{x}^* = \mathbf{d}$, Eq. (4.10) becomes

$$f(\mathbf{x}^* + \mathbf{d}) = f(\mathbf{x}^*) + \nabla f^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \mathbf{H} \mathbf{d} + R \quad (4.11)$$

Often a change in the function is desired when \mathbf{x}^* moves to a neighboring point \mathbf{x} . Defining the change as $\Delta f = f(\mathbf{x}) - f(\mathbf{x}^*)$, Eq. (4.11) gives

$$\Delta f = \nabla f^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \mathbf{H} \mathbf{d} + R \quad (4.12)$$

A first-order change in $f(\mathbf{x})$ at \mathbf{x}^* (denoted as δf) is obtained by retaining only the first term in Eq. (4.12),

$$\delta f = \nabla f^T \delta \mathbf{x} \quad (4.13)$$

where $\delta \mathbf{x}$ is a small change in \mathbf{x}^* ($\delta \mathbf{x} = \mathbf{x} - \mathbf{x}^*$). Note that the first-order change of the function given in Eq. (4.13) is simply a dot product of the vectors ∇f and $\delta \mathbf{x}$. A first-order change is an acceptable approximation for change in the original function when \mathbf{x} is near \mathbf{x}^* .

In Examples 4.7 to 4.9, we now consider some functions and approximate them at the given point \mathbf{x}^* using Taylor's expansion. The remainder R will be dropped while using Eq. (4.11).

EXAMPLE 4.7 Taylor's Expansion of a Function of One Variable

Approximate $f(x) = \cos x$ around the point $x^* = 0$.

Solution. Derivatives of the function $f(x)$ are given as

$$\frac{df}{dx} = -\sin x, \quad \frac{d^2f}{dx^2} = -\cos x \quad (a)$$

Therefore, using Eq. (4.6), the second-order Taylor's expansion for $\cos x$ at the point $x^* = 0$ is given as

$$\cos x \approx \cos 0 - \sin 0(x - 0) + \frac{1}{2}(-\cos 0)(x - 0)^2 \approx 1 - \frac{1}{2}x^2 \quad (b)$$

EXAMPLE 4.8 Taylor's Expansion of a Function of Two Variables

Obtain second-order Taylor's expansion for the function $f(\mathbf{x}) = 3x_1^3x_2$ at the point $\mathbf{x}^* = (1, 1)$:

Solution. The gradient and Hessian of the function $f(\mathbf{x})$ at the point $\mathbf{x}^* = (1, 1)$ using Eqs. (4.3) and (4.5) are

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 9x_1^2x_2 \\ 3x_1^3 \end{bmatrix} = \begin{bmatrix} 9 \\ 3 \end{bmatrix}; \quad \mathbf{H} = \begin{bmatrix} 18x_1x_2 & 9x_1^2 \\ 9x_1^2 & 0 \end{bmatrix} = \begin{bmatrix} 18 & 9 \\ 9 & 0 \end{bmatrix} \quad (\text{a})$$

Substituting these in the matrix form of Taylor's expression given in Eq. (4.10), and using $\mathbf{d} = \mathbf{x} - \mathbf{x}^*$, we obtain an approximation $\tilde{f}(\mathbf{x})$ for $f(\mathbf{x})$ as

$$\tilde{f}(\mathbf{x}) = 3 + \begin{bmatrix} 9 \\ 3 \end{bmatrix}^T \begin{bmatrix} (x_1 - 1) \\ (x_2 - 1) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} (x_1 - 1) \\ (x_2 - 1) \end{bmatrix}^T \begin{bmatrix} 18 & 9 \\ 9 & 0 \end{bmatrix} \begin{bmatrix} (x_1 - 1) \\ (x_2 - 1) \end{bmatrix} \quad (\text{b})$$

where $f(\mathbf{x}^*) = 3$ has been used. Simplifying the expression by expanding vector and matrix products, we obtain Taylor's expansion for $f(\mathbf{x})$ about the point $(1, 1)$ as

$$\tilde{f}(\mathbf{x}) = 9x_1^2 + 9x_1x_2 - 18x_1 - 6x_2 + 9 \quad (\text{c})$$

This expression is a second-order approximation of the function $3x_1^3x_2$ about the point $\mathbf{x}^* = (1, 1)$. That is, in a small neighborhood of \mathbf{x}^* , the expression will give almost the same value as the original function $f(\mathbf{x})$. To see how accurately $\tilde{f}(\mathbf{x})$ approximates $f(\mathbf{x})$, we evaluate these functions for a 30 percent change in the given point $(1, 1)$; i.e., at the point $(1.3, 1.3)$ as $\tilde{f}(\mathbf{x}) = 8.2200$ and $f(\mathbf{x}) = 8.5683$. Therefore the approximate function underestimates the original function by only 4 percent. This is quite a reasonable approximation for many practical applications.

EXAMPLE 4.9 Linear Taylor's Expansion of a Function

Obtain linear Taylor's expansion for the function

$$f(\mathbf{x}) = x_1^2 + x_2^2 - 4x_1 - 2x_2 + 4 \quad (\text{a})$$

at the point $\mathbf{x}^* = (1, 2)$. Compare the approximate function with the original function in a neighborhood of the point $(1, 2)$.

Solution. The gradient of the function at the point $(1, 2)$ is given as

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} (2x_1 - 4) \\ (2x_2 - 2) \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix} \quad (\text{b})$$

Since $f(1, 2) = 1$, Eq. (4.10) gives linear Taylor series approximation for $f(\mathbf{x})$ as

$$\bar{f}(\mathbf{x}) = 1 + [-2 \quad 2] \begin{bmatrix} (x_1 - 1) \\ (x_2 - 2) \end{bmatrix} = -2x_1 + 2x_2 - 1 \quad (c)$$

To see how accurately $\bar{f}(\mathbf{x})$ approximates the original $f(\mathbf{x})$ in the neighborhood of (1, 2), we calculate the functions at the point (1.1, 2.2), a 10 percent change in the point as $\bar{f}(\mathbf{x}) = 1.20$ and $f(\mathbf{x}) = 1.25$. We see that the approximate function underestimates the real function by 4 percent. An error of this magnitude is quite acceptable in many applications. Note, however, that the errors will be different for different functions and can be larger for highly nonlinear functions.

4.2.4 Quadratic Forms and Definite Matrices

Quadratic Form Quadratic form is a special nonlinear function having only second-order terms, e.g., the function

$$F(\mathbf{x}) = x_1^2 + 2x_2^2 + 3x_3^2 + 2x_1x_2 + 2x_2x_3 + 2x_3x_1$$

Quadratic forms play a prominent role in optimization theory and methods. Therefore, in this subsection, we discuss some results related to them. Consider a special function of n variables $F(\mathbf{x}) = F(x_1, x_2, \dots, x_n)$ written in the double summation notation as (refer to Section 1.5 for the summation notation):

$$F(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n p_{ij} x_i x_j \quad (4.14)$$

where p_{ij} are known constants and the factor $\frac{1}{2}$ is used for convenience only to match $F(\mathbf{x})$ with the second-order term in the Taylor series expansion of Eq. (4.9). The results of this section will not be affected if the factor $\frac{1}{2}$ is not used, as is done in many other texts. Only the value of the quadratic form is affected by the factor. Expanding Eq. (4.14) by setting $i = 1$ and letting j vary from 1 to n , and then setting $i = 2$ and letting j vary again from 1 to n , and so on, we get

$$\begin{aligned} F(\mathbf{x}) = \frac{1}{2} [& (p_{11}x_1^2 + p_{12}x_1x_2 + \dots + p_{1n}x_1x_n) \\ & + (p_{21}x_2x_1 + p_{22}x_2^2 + \dots + p_{2n}x_2x_n) \\ & + \dots + (p_{n1}x_nx_1 + \dots + p_{nn}x_n^2)] \end{aligned} \quad (4.15)$$

*Note that with coefficients p_{ij} specified and the variables x_i given, $F(\mathbf{x})$ in Eq. (4.15) is just a number (scalar). The function is called a *quadratic form* because each of its terms is either the square of a variable or product of two different variables.*

Matrix of the Quadratic Form The quadratic form can be written in the matrix notation. Let $\mathbf{P} = [p_{ij}]$ be an $n \times n$ matrix, $\mathbf{x} = (x_1, x_2, \dots, x_n)$ an n -dimensional vector, and $\mathbf{y} = (y_1, y_2, \dots, y_n)$ another n dimensional vector obtained by multiplying \mathbf{P} by \mathbf{x} . Writing $\mathbf{y} = \mathbf{P}\mathbf{x}$ in the summation notation we get

$$y_i = \sum_{j=1}^n p_{ij} x_j; \quad i = 1 \text{ to } n \quad (4.16)$$

Also, we can rewrite Eq. (4.14) as

$$F(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^n x_i \left(\sum_{j=1}^n p_{ij} x_j \right) \quad (4.17)$$

Substituting Eq. (4.16) into Eq. (4.17),

$$F(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^n x_i y_i \quad (4.18)$$

But, the summation on the right side of Eq. (4.18) represents the scalar product of vectors \mathbf{x} and \mathbf{y} as $\mathbf{x}^T \mathbf{y}$. Substituting $\mathbf{y} = \mathbf{P}\mathbf{x}$ in this, we obtain the matrix representation for $F(\mathbf{x})$

$$F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{y} = \frac{1}{2} \mathbf{x}^T \mathbf{P}\mathbf{x} \quad (4.19)$$

\mathbf{P} is called the *matrix of the quadratic form* $F(\mathbf{x})$. Elements of \mathbf{P} are identified as coefficients of the terms in the function $F(\mathbf{x})$. For example, in Eq. (4.15) element p_{ij} is twice the coefficient of the term $x_i x_j$ in $F(\mathbf{x})$. We see that except for the squared terms, each product $x_i x_j (i \neq j)$ appears twice. Therefore, Eq. (4.15) can be rewritten as

$$\begin{aligned} F(\mathbf{x}) = \frac{1}{2} \{ & [p_{11}x_1^2 + p_{22}x_2^2 + \dots + p_{nn}x_n^2] \\ & + [(p_{12} + p_{21})x_1x_2 + (p_{13} + p_{31})x_1x_3 + \dots + (p_{1n} + p_{n1})x_1x_n] \\ & + [(p_{23} + p_{32})x_2x_3 + (p_{24} + p_{42})x_2x_4 + \dots + (p_{2n} + p_{n2})x_2x_n] \\ & + \dots + [(p_{n-1,n} + p_{n,n-1})x_{n-1}x_n] \} \end{aligned} \quad (4.20)$$

Thus, the coefficient of $x_i x_j$ is $\frac{1}{2}(p_{ij} + p_{ji})$ for $j > i$. Define coefficients of an $n \times n$ matrix \mathbf{A} as

$$a_{ij} = \frac{1}{2}(p_{ij} + p_{ji}), \quad \text{for all } i \text{ and } j \quad (4.21)$$

Using this definition, it can be easily seen that

$$a_{ij} + a_{ji} = p_{ij} + p_{ji}$$

Therefore, $(p_{ij} + p_{ji})$ in Eq. (4.20) can be replaced with $(a_{ij} + a_{ji})$ and the quadratic form of Eq. (4.19) becomes

$$F(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{P}\mathbf{x} = \frac{1}{2} \mathbf{x}^T \mathbf{A}\mathbf{x} \quad (4.22)$$

The value of the quadratic form does not change with \mathbf{P} replaced by \mathbf{A} . The matrix \mathbf{A} , however, is always *symmetric* ($a_{ij} = a_{ji}$) whereas \mathbf{P} is generally not. Symmetry of \mathbf{A} can be easily seen from the definition of a_{ij} given in Eq. (4.21), i.e., interchanging the indices i and j , we get $a_{ji} = a_{ij}$. Thus, given any quadratic form $\frac{1}{2}\mathbf{x}^T \mathbf{P}\mathbf{x}$ we can always replace \mathbf{P} with a symmetric matrix. The preceding discussion also shows that many matrices can be associ-

ated with the same quadratic form. All of them are asymmetric except one. The symmetric matrix associated with it is always unique. Asymmetric matrices are not very useful. The symmetric matrix, however, determines the nature of the quadratic form, which will be discussed later in this section. Comparing Eq. (4.11) with Eq. (4.22), we observe that the third term of Taylor's expansion is a quadratic form in the variables \mathbf{d} . Therefore, the Hessian \mathbf{H} is a matrix associated with that quadratic form. Example 4.10 illustrates identification of matrices associated with a quadratic form.

EXAMPLE 4.10 Matrix of the Quadratic Form

Identify a matrix associated with the quadratic form

$$F(x_1, x_2, x_3) = \frac{1}{2}(2x_1^2 + 2x_1x_2 + 4x_1x_3 - 6x_2^2 - 4x_2x_3 + 5x_3^2) \quad (\text{a})$$

Solution. Writing F in the matrix form ($F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{P}\mathbf{x}$), we obtain

$$F(\mathbf{x}) = \frac{1}{2} \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 2 & 2 & 4 \\ 0 & -6 & -4 \\ 0 & 0 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (\text{b})$$

The matrix \mathbf{P} of the quadratic form can be easily identified by comparing the expression with Eq. (4.20). The i th diagonal element p_{ii} is the coefficient of x_i^2 . Therefore, $p_{11} = 2$, the coefficient of x_1^2 ; $p_{22} = -6$, the coefficient of x_2^2 ; and $p_{33} = 5$, the coefficient of x_3^2 . The coefficient of $x_i x_j$ can be divided in any proportion between the elements p_{ij} and p_{ji} of the matrix \mathbf{P} as long as the sum $p_{ij} + p_{ji}$ is equal to the coefficient of $x_i x_j$. In the above matrix $p_{12} = 2$ and $p_{21} = 0$, giving $p_{12} + p_{21} = 2$, which is the coefficient of $x_1 x_2$. Similarly, we can calculate the elements p_{13} , p_{31} , p_{23} , and p_{32} .

Since the coefficient of $x_i x_j$ can be divided between p_{ij} and p_{ji} in any proportion, there are many matrices associated with a quadratic form. For example, the following matrices are also associated with the same quadratic form:

$$\begin{bmatrix} 2 & 0.5 & 1 \\ 1.5 & -6 & -6 \\ 3 & 2 & 5 \end{bmatrix}; \begin{bmatrix} 2 & 4 & 5 \\ -2 & -6 & 4 \\ -1 & -8 & 5 \end{bmatrix} \quad (\text{c})$$

Dividing the coefficients equally between p_{ij} and p_{ji} , we obtain

$$F(\mathbf{x}) = \frac{1}{2} \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 2 & 1 & 2 \\ 1 & -6 & -2 \\ 2 & -2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (\text{d})$$

Any of the matrices in Eqs. (b) to (d) give a matrix associated with the quadratic form. However, the matrix in Eq. (d) is symmetric. The diagonal elements of the symmetric matrix are obtained from the coefficient of x_i^2 as before. The off-diagonal elements are obtained by dividing the coefficient of the term $x_i x_j$ equally between a_{ij} and a_{ji} . This satisfies Eq. (4.21).

Form of a Matrix Quadratic form $F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}$ may be either positive, negative, or zero for any \mathbf{x} . It may also have the property of being always positive for any \mathbf{x} [except for $F(\mathbf{0})$]. Such a form is called positive definite. Similarly, it is called *negative definite* if $\mathbf{x}^T\mathbf{A}\mathbf{x} < 0$ for all \mathbf{x} except $\mathbf{x} = \mathbf{0}$. If a quadratic form has the property $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ for all \mathbf{x} and there exists at least one $\mathbf{x} \neq \mathbf{0}$ (nonzero \mathbf{x}) with $\mathbf{x}^T\mathbf{A}\mathbf{x} = 0$, then it is called *positive semidefinite*. A similar definition for *negative semidefinite* is obtained by reversing the sense of the inequality. A quadratic form that is positive for some vectors \mathbf{x} and negative for others is called *indefinite*. A symmetric matrix \mathbf{A} is often referred to as a *positive definite*, *positive semidefinite*, *negative definite*, *negative semidefinite*, or *indefinite* if the quadratic form associated with \mathbf{A} is positive definite, positive semidefinite, negative definite, negative semidefinite, or indefinite, respectively (Example 4.11).

EXAMPLE 4.11 Determination of the Form of a Matrix

Determine the form of the following matrices:

$$(i) \mathbf{A} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 3 \end{bmatrix} \quad (ii) \mathbf{A} = \begin{bmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Solution. The quadratic form associated with the matrix (i) is always positive, i.e.,

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = (2x_1^2 + 4x_2^2 + 3x_3^2) > 0 \quad (a)$$

unless $x_1 = x_2 = x_3 = 0$ ($\mathbf{x} = \mathbf{0}$). Thus, the matrix is positive definite. The quadratic form associated with the matrix (ii) is negative semidefinite, since

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = (-x_1^2 - x_2^2 + 2x_1x_2 - x_3^2) = \{-x_3^2 - (x_1 - x_2)^2\} \leq 0 \quad (b)$$

for all \mathbf{x} , and $\mathbf{x}^T\mathbf{A}\mathbf{x} = 0$ when $x_3 = 0$, and $x_1 = x_2$ [e.g., $\mathbf{x} = (1, 1, 0)$]. The quadratic form is not negative definite but is negative semidefinite since it can have zero value for nonzero \mathbf{x} . Therefore, the matrix associated with it is also negative semidefinite.

We will now discuss *methods for checking* positive definiteness or semidefiniteness (form) of a quadratic form or a matrix. Since this involves calculation of eigenvalues of a matrix, Section B.6 in Appendix B should be reviewed at this point.

Theorem 4.2 Eigenvalue Check for the Form of a Matrix Let λ_i , $i = 1$ to n be n eigenvalues of a symmetric $n \times n$ matrix \mathbf{A} associated with the quadratic form $F(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}$ (since \mathbf{A} is symmetric, all eigenvalues are real). The following results can be stated regarding the quadratic form $F(\mathbf{x})$ or the matrix \mathbf{A} :

1. $F(\mathbf{x})$ is a *positive definite* if and only if all eigenvalues of \mathbf{A} are strictly positive, i.e., $\lambda_i > 0$, $i = 1$ to n .
2. $F(\mathbf{x})$ is *positive semidefinite* if and only if all eigenvalues of \mathbf{A} are nonnegative, i.e., $\lambda_i \geq 0$, $i = 1$ to n (note that at least one eigenvalue must be zero for it to be called positive semidefinite).

3. $F(\mathbf{x})$ is *negative definite* if and only if all eigenvalues of \mathbf{A} are strictly negative, i.e., $\lambda_i < 0$, $i = 1$ to n .
4. $F(\mathbf{x})$ is *negative semidefinite* if and only if all eigenvalues of \mathbf{A} are nonpositive, i.e., $\lambda_i \leq 0$, $i = 1$ to n (note that at least one eigenvalue must be zero for it to be called negative semidefinite).
5. $F(\mathbf{x})$ is *indefinite* if some $\lambda_i < 0$ and some other $\lambda_i > 0$.

Another way of checking the form of a matrix is provided by the following theorem:

Theorem 4.3 Check for the Form of a Matrix Using Principal Minors Let M_k be the k th leading principal minor of the $n \times n$ symmetric matrix \mathbf{A} defined as the determinant of a $k \times k$ submatrix obtained by deleting the last $(n - k)$ rows and columns of \mathbf{A} (Appendix B, Section B.3). Assume that *no two consecutive principal minors* are zero. Then

1. \mathbf{A} is *positive definite* if and only if all $M_k > 0$, $k = 1$ to n .
2. \mathbf{A} is *positive semidefinite* if and only if $M_k > 0$, $k = 1$ to r , where $r < n$ is the rank of \mathbf{A} (refer to Appendix B, Section B.4 for definition of rank of a matrix).
3. \mathbf{A} is *negative definite* if and only if $M_k < 0$ for k odd and $M_k > 0$ for k even, $k = 1$ to n .
4. \mathbf{A} is *negative semidefinite* if and only if $M_k < 0$ for k odd and $M_k > 0$ for k even, $k = 1$ to $r < n$.
5. \mathbf{A} is *indefinite* if it does not satisfy any of the preceding criteria.

This theorem is applicable only if the assumption of no two consecutive principal minors being zero is satisfied. When there are consecutive zero principal minors, we may resort to the eigenvalue check of Theorem 4.2. Note also that a *positive definite matrix cannot have negative or zero diagonal elements*. The form of a matrix is determined in Example 4.12.

The theory of quadratic forms is used in second-order conditions for a local optimum point. Also, it is used to determine convexity of functions of the optimization problem. Convex functions play a role in determining the global optimum point. These topics are discussed in later sections.

EXAMPLE 4.12 Determination of the Form of a Matrix

Determine the form of the matrices given in Example 4.11.

Solution. For a given matrix \mathbf{A} , the eigenvalue problem is defined as $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, where λ is an eigenvalue and \mathbf{x} is the corresponding eigenvector (refer to Section B.6 in Appendix B for more details). To determine the eigenvalues, we set the so-called characteristic determinant to zero $|\mathbf{A} - \lambda\mathbf{I}| = 0$. Since the matrix (i) is diagonal, its eigenvalues are the diagonal elements, i.e., $\lambda_1 = 2$, $\lambda_2 = 3$, and $\lambda_3 = 4$. Since all eigenvalues are strictly positive, the matrix is positive definite. The principal minor check of Theorem 4.3 also gives the same conclusion.

For the matrix (ii), the characteristic determinant of the eigenvalue problem is

$$\begin{vmatrix} -1-\lambda & 1 & 0 \\ 0 & -1-\lambda & 0 \\ 0 & 0 & -1-\lambda \end{vmatrix} = 0 \quad (\text{a})$$

Expanding the determinant by the third row, we obtain

$$(-1 - \lambda)[(-1 - \lambda)^2 - 1] = 0 \quad (b)$$

Therefore, the three roots give the eigenvalues as $\lambda_1 = -2$, $\lambda_2 = -1$, and $\lambda_3 = 0$. Since all eigenvalues are nonpositive, the matrix is negative semidefinite. To use Theorem 4.3, we calculate the three leading principal minors as

$$M_1 = -1, \quad M_2 = \begin{vmatrix} -1 & 1 \\ 1 & -1 \end{vmatrix} = 0, \quad M_3 = \begin{vmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & -1 \end{vmatrix} = 0 \quad (c)$$

Since there are two consecutive zero leading principal minors, we cannot use Theorem 4.3.

Differentiation of a Quadratic Form On several occasions we would like to find gradient and Hessian matrix for the quadratic form. We consider the symmetric quadratic form of Eq. (4.22) and write it in summation notation as

$$F(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i x_j \quad (4.23)$$

To calculate derivatives of $F(\mathbf{x})$, we first expand the summations and then differentiate the expression with respect to x_i to obtain

$$\frac{\partial F(\mathbf{x})}{\partial x_i} = \sum_{j=1}^n a_{ij} x_j \quad (4.24)$$

Writing the partial derivatives of Eq. (4.24) in a column vector, we get the *gradient of the quadratic form* as

$$\nabla F(\mathbf{x}) = \mathbf{A}\mathbf{x} \quad (4.25)$$

Differentiating Eq. (4.24) once again with respect to x_i we get

$$\frac{\partial^2 F(\mathbf{x})}{\partial x_j \partial x_i} = a_{ij} \quad (4.26)$$

Equation (4.26) shows that the components a_{ij} of the matrix \mathbf{A} are the components of the *Hessian matrix for the quadratic form*. Example 4.13 shows the calculations for the gradient and Hessian of the quadratic form.

EXAMPLE 4.13 Calculations for the Gradient and Hessian of the Quadratic Form

Calculate the gradient and Hessian of the following quadratic form:

$$F(\mathbf{x}) = \frac{1}{2}(2x_1^2 + 2x_1x_2 + 4x_1x_3 - 6x_2^2 - 4x_2x_3 + 5x_3^2) \quad (\text{a})$$

Solution. Differentiating $F(\mathbf{x})$ with respect to x_1 , x_2 , and x_3 , we get gradient components as

$$\frac{\partial F}{\partial x_1} = 2x_1 + x_2 + 2x_3; \quad \frac{\partial F}{\partial x_2} = x_1 - 6x_2 - 2x_3; \quad \frac{\partial F}{\partial x_3} = 2x_1 - 2x_2 + 5x_3 \quad (\text{b})$$

Differentiating the gradient components once again, we get the Hessian components as

$$\begin{aligned} \frac{\partial^2 F}{\partial x_1^2} &= 2, & \frac{\partial^2 F}{\partial x_1 \partial x_2} &= 1, & \frac{\partial^2 F}{\partial x_1 \partial x_3} &= 2 \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} &= 1, & \frac{\partial^2 F}{\partial x_2^2} &= -6, & \frac{\partial^2 F}{\partial x_2 \partial x_3} &= -2 \\ \frac{\partial^2 F}{\partial x_3 \partial x_1} &= 2, & \frac{\partial^2 F}{\partial x_3 \partial x_2} &= -2, & \frac{\partial^2 F}{\partial x_3^2} &= 5 \end{aligned} \quad (\text{c})$$

Writing the given quadratic form in a matrix form, we identify matrix \mathbf{A} as

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 2 \\ 1 & -6 & -2 \\ 2 & -2 & 5 \end{bmatrix} \quad (\text{d})$$

Comparing elements of the matrix \mathbf{A} with second partial derivatives of F , we observe that the Hessian $\mathbf{H} = \mathbf{A}$. Using Eq. (4.25), the gradient of the quadratic form is also given as

$$\nabla F(\mathbf{x}) = \begin{bmatrix} 2 & 1 & 2 \\ 1 & -6 & -2 \\ 2 & -2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} (2x_1 + x_2 + 2x_3) \\ (x_1 - 6x_2 - 2x_3) \\ (2x_1 - 2x_2 + 5x_3) \end{bmatrix} \quad (\text{e})$$

4.2.5 Concept of Necessary and Sufficient Conditions

In the remainder of this chapter, we shall describe necessary and sufficient conditions for optimality of unconstrained and constrained optimization problems. It is important to understand the meaning of the terms *necessary* and *sufficient*. These terms have general meaning in mathematical analyses. We shall, however, discuss them for the optimization problem only. The optimality conditions are derived by assuming that we are at an optimum point and then studying the behavior of the functions and their derivatives at the point. *The conditions that must be satisfied at the optimum point are called necessary. Stated differently, if a point does not satisfy*

the necessary conditions, it cannot be optimum. Note, however, that satisfaction of necessary conditions does not guarantee optimality of the point, i.e., there can be nonoptimum points that also satisfy the same conditions. This indicates that the number of points satisfying necessary conditions can be more than the number of optima. Points satisfying the necessary conditions are called *candidate optimum points*. We must, therefore, perform further tests to distinguish between optimum and nonoptimum points, both satisfying the necessary conditions.

The *sufficient conditions* provide tests to distinguish between optimum and nonoptimum points. *If a candidate optimum point satisfies the sufficient conditions, then it is indeed optimum.* We do not need any further tests. If the sufficient conditions are not satisfied, however, or cannot be used, we may not be able to conclude that the candidate design is not optimum. Our conclusion will depend on the assumptions and restrictions used in deriving the sufficient conditions. Further analysis of the problem or other conditions are needed to make a definite statement about optimality of the candidate point. *In summary,*

1. Optimum points must satisfy the necessary conditions. Points that do not satisfy them cannot be optimum.
2. A point satisfying the necessary conditions need not be optimum, i.e., nonoptimum points may also satisfy the necessary conditions.
3. A candidate point satisfying a sufficient condition is indeed optimum.
4. If sufficiency conditions cannot be used or they are not satisfied, we may not be able to draw any conclusions about optimality of the candidate point.

4.3 Unconstrained Optimum Design Problems

We are now ready to discuss the theory and concepts of optimum design. In this section, we shall discuss necessary and sufficient conditions for unconstrained optimization problems defined as: Minimize $f(\mathbf{x})$ without any constraints on \mathbf{x} . Such problems arise infrequently in practical engineering applications. However, we consider them here because optimality conditions for constrained problems are a logical extension of these conditions. In addition, one numerical strategy for solving a constrained problem is to convert it into a sequence of unconstrained problems. Thus, it is important to completely understand unconstrained optimization concepts.

The optimality conditions for unconstrained or constrained problems can be used in two ways:

1. The optimality conditions can be used to check whether a given point is a local optimum for the problem.
2. The optimality conditions can be solved for local optimum points.

We will discuss only the *local optimality conditions* for unconstrained problems. Global optimality will be discussed in Section 4.6. First the necessary and then the sufficient conditions will be discussed. The necessary conditions must be satisfied at the minimum point, otherwise it cannot be a minimum. These conditions, however, may also be satisfied by a point that is not minimum. A point satisfying the necessary conditions is simply a candidate local minimum. The sufficient conditions distinguish minimum points from others. We shall elaborate these concepts further with some examples.

4.3.1 Concepts Related to Optimality Conditions

The basic concept for obtaining local optimality conditions is to assume that we are at a minimum point \mathbf{x}^* and then examine a small neighborhood to study properties of the function and its derivatives. Basically, we use the definition of a local minimum given in Inequality (4.1) to derive the optimality conditions. Since we examine only a small neighborhood, the conditions we obtain are called *local*.

Let \mathbf{x}^* be a *local minimum* point for $f(\mathbf{x})$. To investigate its neighborhood, let \mathbf{x} be any point near \mathbf{x}^* . Define increments \mathbf{d} and Δf in \mathbf{x}^* and $f(\mathbf{x}^*)$, as $\mathbf{d} = \mathbf{x} - \mathbf{x}^*$ and $\Delta f = f(\mathbf{x}) - f(\mathbf{x}^*)$. Since $f(\mathbf{x})$ has a local minimum at \mathbf{x}^* , it will not reduce any further if we move a small distance away. Therefore, a change in the function for any move in a small neighborhood of \mathbf{x}^* must be nonnegative, i.e., the function value must either remain constant or increase. This condition, also obtained directly from the definition of local minimum given in Inequality (4.1), can be expressed as the following inequality:

$$\Delta f = f(\mathbf{x}) - f(\mathbf{x}^*) \geq 0 \quad (4.27)$$

for all small changes \mathbf{d} . The inequality in Eq. (4.27) can be used to derive necessary and sufficient conditions for a local minimum point. Since \mathbf{d} is small, we can approximate Δf by Taylor's expansion at \mathbf{x}^* and derive optimality conditions using it.

4.3.2 Optimality Conditions for Functions of Single Variable

First-Order Necessary Conditions Let us first consider only a *function of one variable*. The Taylor's expansion of $f(x)$ at the point x^* gives

$$f(x) = f(x^*) + f'(x^*)d + \frac{1}{2}f''(x^*)d^2 + R$$

where R is the remainder containing higher-order terms in d and "primes" indicate the order of the derivatives. From this equation, the change in the function at x^* , i.e., $\Delta f = f(x) - f(x^*)$, is given as

$$\Delta f(x) = f'(x^*)d + \frac{1}{2}f''(x^*)d^2 + R \quad (4.28)$$

The Inequality (4.27) shows that expression for Δf must be nonnegative as (≥ 0) as x^* is a local minimum. Since d is small, the first-order term $f'(x^*)d$ dominates other terms and therefore Δf can be approximated as $\Delta f \approx f'(x^*)d$. Note that Δf in this equation can be positive or negative depending on the sign of the term $f'(x^*)d$. Since d is an arbitrary, small increment in x^* , it may be positive or negative. Therefore, if $f'(x^*) \neq 0$, the term $f'(x^*)d$ (and hence Δf) can be negative. To see this more clearly, let the term be positive for some increment d_1 that satisfies the Inequality (4.27), i.e., $\Delta f = f'(x^*)d_1 > 0$. Since the increment d is arbitrary, it is reversible, so $d_2 = -d_1$ is another possible increment. For d_2 , Δf becomes negative, which violates the Inequality (4.27). Thus, the quantity $f'(x^*)d$ can have a negative value regardless of the sign of $f'(x^*)$, unless it is zero. The only way it can be nonnegative for all d in a neighborhood of x^* is when

$$f'(x^*) = 0 \quad (4.29)$$

Equation (4.29) is a *first-order necessary condition* for the local minimum of $f(x)$ at x^* . It is called "first-order" because it only involves the first derivative of the function. Note that the preceding arguments can be used to show that the condition of Eq. (4.29) is also necessary for local maximum points. Therefore, since the points satisfying Eq. (4.29) can be local minima, maxima, or neither minimum nor maximum (*inflection points*), they are called *stationary points*.

Sufficient Conditions Now we need a *sufficient condition* to determine which of the stationary points are actually minimum for the function. Since stationary points satisfy the necessary condition $f'(x^*) = 0$, the change in function Δf of Eq. (4.28) becomes

$$\Delta f(x) = \frac{1}{2} f''(x^*)d^2 + R \quad (4.30)$$

Since the second-order term dominates all other higher-order terms, we need to focus on it. Note that the term can be positive for all $d \neq 0$, if

$$f''(x^*) > 0 \quad (4.31)$$

Stationary points satisfying Inequality (4.31) must be at least local minima because they satisfy Inequality (4.27) ($\Delta f > 0$). That is, the function has positive curvature at the minimum points. Inequality (4.31) is then *sufficient* for x^* to be a local minimum. Thus, if we have a point x^* satisfying both conditions in Eqs. (4.29) and (4.31), then any small move away from it will either increase the function value or keep it unchanged. This indicates that $f(x^*)$ has the smallest value in a small neighborhood (local minimum) of the point x^* . Note that the foregoing conditions can be stated in terms of the *curvature of the function* since second derivative is the curvature.

Second-Order Necessary Condition If Inequality (4.31) is not satisfied [e.g., $f''(x^*) = 0$], we cannot conclude that x^* is not a minimum point. Note, however, from Eqs. (4.27) and (4.28) that $f(x^*)$ cannot be a minimum unless

$$f''(x^*) \geq 0 \quad (4.32)$$

That is, if f'' evaluated at the candidate point x^* is less than zero, then x^* is not a local minimum point. Inequality (4.32) is known as a *second-order necessary condition*, so any point violating it [i.e., $f''(x^*) < 0$] cannot be a local minimum.

If $f''(x^*) = 0$, we need to evaluate higher-order derivatives to determine if the point is a local minimum (see Examples 4.14 to 4.18). By the arguments used to derive Eq. (4.29), $f'''(x^*)$ must be zero for the stationary point (necessary condition) and $f^{IV}(x^*) > 0$ for x^* to be a local minimum. In general, *the lowest nonzero derivative must be even-ordered for stationary points (necessary conditions), and it must be positive for local minimum points (sufficiency condition). All odd-ordered derivatives lower than the nonzero even-ordered derivative must be zero as the necessary condition.*

EXAMPLE 4.14 Determination of Local Minimum Points Using Necessary Conditions

Find local minima for the function $f(x) = \sin x$.

Solution. Differentiating the function twice,

$$f' = \cos x; \quad f'' = -\sin x; \quad (a)$$

Stationary points are obtained as roots of $f'(x) = 0$ ($\cos x = 0$). These are

$$x = \pm\pi/2, \pm3\pi/2, \pm5\pi/2, \pm7\pi/2, \dots \quad (b)$$

Local minima are identified as

$$x^* = 3\pi/2, 7\pi/2, \dots; \quad -\pi/2, -5\pi/2, \dots \quad (c)$$

since these points satisfy the sufficiency condition of Eq. (4.31) ($f'' = -\sin x > 0$). The minimum value of $\sin x$ at the points x^* is -1 . This is true from the graph of the function $\sin x$. There are infinite minimum points, and they are all actually global minima. The points $\pi/2, 5\pi/2, \dots$, and $-3\pi/2, -7\pi/2, \dots$ are global maximum points where $\sin x$ has a value of 1. At these points, $f'(x) = 0$ and $f''(x) < 0$.

EXAMPLE 4.15 Determination of Local Minimum Points Using Necessary Conditions

Find local minima for the function $f(x) = x^2 - 4x + 4$.

Solution. Figure 4-7 shows a graph for the function $f(x) = x^2 - 4x + 4$. It can be seen that the function always has a positive value except at $x = 2$, where it is zero. Therefore, this is a local as well as global minimum point for the function. Let us see how this point will be determined using the necessary and sufficient conditions.

Differentiating the function twice,

$$f' = 2x - 4; \quad f'' = 2 \tag{a}$$

The necessary condition $f' = 0$ implies that $x^* = 2$ is a stationary point. Since $f'' > 0$ at $x^* = 2$ (actually for all x), the sufficiency condition of Eq. (4.31) is satisfied. Therefore $x^* = 2$ is a local minimum for $f(x)$. The minimum value of f is 0 at $x^* = 2$. Note that at $x^* = 2$, the second-order necessary condition for a local maximum $f'' \leq 0$ is violated since $f''(2) = 2 > 0$. Therefore the point $x^* = 2$ cannot be a local maximum point. In fact the graph of the function shows that there is no local or global maximum point for the function.

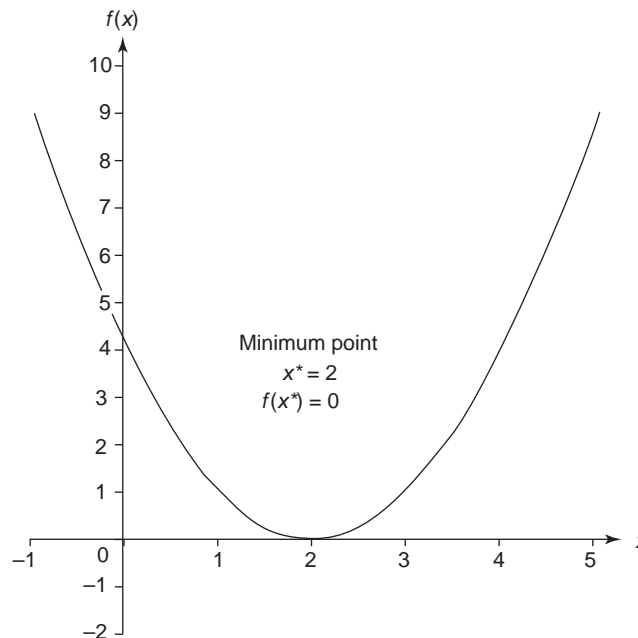


FIGURE 4-7 Graph of $f(x) = x^2 - 4x + 4$ of Example 4.15.

EXAMPLE 4.16 Determination of Local Minimum Points Using Necessary Conditions

Find local minima for the function $f(x) = x^3 - x^2 - 4x + 4$.

Solution. Figure 4-8 shows the graph of the function. It can be seen that point A is a local minimum point and point B is a local maximum point. We shall use the necessary and sufficient conditions to prove that this is indeed true. Differentiating the function,

$$f' = 3x^2 - 2x - 4; \quad f'' = 6x - 2 \quad (\text{a})$$

For this example there are two points satisfying the necessary condition of Eq. (4.29), i.e., stationary points. These are obtained as roots of the equation $f'(x) = 0$,

$$x_1^* = \frac{1}{6}(2 + 7.211) = 1.535 \quad (\text{Point A}) \quad (\text{b})$$

$$x_2^* = \frac{1}{6}(2 - 7.211) = -0.8685 \quad (\text{Point B}) \quad (\text{c})$$

Evaluating f'' at these points,

$$f''(1.535) = 7.211 > 0$$

$$f''(-0.8685) = -7.211 < 0 \quad (\text{d})$$

We see that only x_1^* satisfies the sufficiency condition ($f'' > 0$) of Eq. (4.31). Therefore, it is a local minimum point. From the graph in Fig. 4-8 we can see that the local minimum $f(x_1^*)$ is not the global minimum. A *global minimum for $f(x)$ does not exist since the domain as well as the function are not bounded* (Theorem 4.1). The value

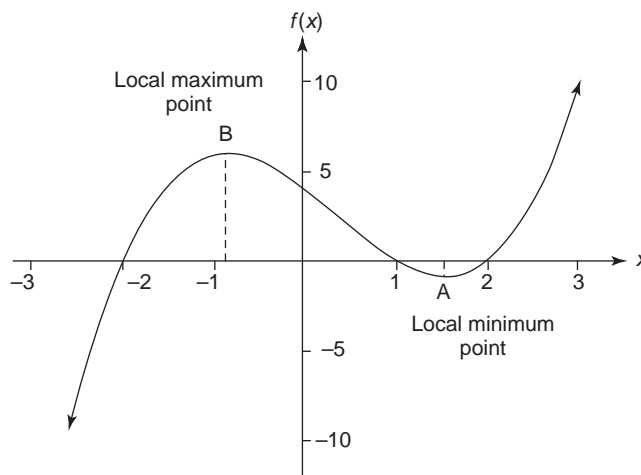


FIGURE 4-8 Graph of $f(x) = x^3 - x^2 - 4x + 4$ of Example 4.16.

of the function at the local minimum is obtained as -0.88 by substituting $x_1^* = 1.535$ in $f(x)$. Note that $x_2^* = -0.8685$ is a local maximum point since $f''(x_2^*) < 0$. The value of the function at the maximum point is 6.065 . There is no global maximum point for the function. Note that the second order necessary condition for a local minimum [$f''(x^*) \geq 0$] is violated at $x_2^* = -0.8685$. Therefore, this stationary point cannot be a local minimum point. Similarly, the stationary point $x_1^* = 1.535$ cannot be a local maximum point.

As noted earlier, the optimality conditions can also be used to check optimality of a given point. To illustrate this, let us check optimality of the point $x = 1$. At this point, $f' = 3(1)^2 - 2(1) - 4 = -3 \neq 0$. Therefore $x = 1$ is not a stationary point and thus cannot be a local minimum or maximum for the function.

EXAMPLE 4.17 Determination of Local Minimum Points Using Necessary Conditions

Find the minimum for the function $f(x) = x^4$.

Solution. Differentiating the function twice,

$$f' = 4x^3; \quad f'' = 12x^2 \quad (a)$$

The necessary condition gives $x^* = 0$ as a stationary point. Since $f'''(x^*) = 0$, we cannot conclude from the sufficiency condition of Eq. (4.31) that x^* is a minimum point. However, the second-order necessary condition of Eq. (4.32) is satisfied, so we cannot rule out the possibility of x^* being a minimum point. In fact, a graph of $f(x)$ versus x will show that x^* is indeed the global minimum point $f''' = 24x$, which is zero at $x^* = 0$. $f^{IV}(x^*) = 24$, which is strictly greater than zero. Therefore, the fourth-order sufficiency condition is satisfied, and $x^* = 0$ is indeed a minimum point. It is actually a global minimum point with $f(0) = 0$.

EXAMPLE 4.18 Minimum Cost Spherical Tank Using Necessary Conditions

The result of a problem formulation in Section 2.3 is a cost function that represents the lifetime cooling related cost of an insulated spherical tank as

$$f(x) = ax + b/x, \quad a, b > 0 \quad (a)$$

where x is the thickness of insulation, and a and b are positive constants.

Solution. To minimize f , we solve the equation (necessary condition)

$$f' = a - b/x^2 = 0 \quad (b)$$

The solution is $x^* = \sqrt{b/a}$. To check if the stationary point is a local minimum, evaluate

$$f''(x^*) = 2b/x^{*3} \quad (c)$$

Since b and x^* are positive, $f''(x^*)$ is positive and x^* is a local minimum point. The value of the function at x^* is $2\sqrt{ab}$. Note that since the function cannot have a negative value because of the physics of the problem, x^* represents a global minimum for the problem.

4.3.3 Optimality Conditions for Functions of Several Variables

For the general case of a function of several variables $f(\mathbf{x})$ where \mathbf{x} is an n -vector, we can repeat the derivation of *necessary and sufficient* conditions using the multidimensional form of Taylor's expansion:

$$f(\mathbf{x}) = f(\mathbf{x}^*) + \nabla f(\mathbf{x}^*)^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \mathbf{H}(\mathbf{x}^*) \mathbf{d} + R$$

Or, change in the function is given as

$$\Delta f = \nabla f(\mathbf{x}^*)^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \mathbf{H}(\mathbf{x}^*) \mathbf{d} + R \quad (4.33)$$

If we assume a local minimum at \mathbf{x}^* then Δf must be nonnegative due to the definition of a local minimum given in Inequality (4.1), i.e., $\Delta f \geq 0$. Concentrating only on the first-order term in Eq. (4.33), we observe (as before) that Δf can be nonnegative for all possible \mathbf{d} when

$$\nabla f(\mathbf{x}^*) = 0 \quad (4.34)$$

That is, the gradient of the function at \mathbf{x}^* must be zero. In the component form, this necessary condition becomes

$$\frac{\partial f(\mathbf{x}^*)}{\partial x_i} = 0; \quad i = 1 \text{ to } n \quad (4.35)$$

Points satisfying Eq. (4.35) are called *stationary points*. Considering the second term in Eq. (4.33) evaluated at a stationary point, the positivity of Δf is assured if

$$\mathbf{d}^T \mathbf{H}(\mathbf{x}^*) \mathbf{d} > 0 \quad (4.36)$$

for all $\mathbf{d} \neq \mathbf{0}$. This will be true if the Hessian $\mathbf{H}(\mathbf{x}^*)$ is a positive definite matrix (see Section 4.2) which is then the sufficient condition for a local minimum of $f(\mathbf{x})$ at \mathbf{x}^* . Conditions (4.35) and (4.36) are the multidimensional equivalent of Conditions (4.29) and (4.31), respectively. We summarize the development of this section in Theorem 4.4.

Theorem 4.4 Necessary and Sufficient Conditions for Local Minimum

Necessary condition. If $f(\mathbf{x})$ has a local minimum at \mathbf{x}^* then

$$\frac{\partial f(\mathbf{x}^*)}{\partial x_i} = 0; \quad i = 1 \text{ to } n \quad (\text{a})$$

Second-order necessary condition. If $f(\mathbf{x})$ has a local minimum at \mathbf{x}^* , then the Hessian matrix of Eq. (4.5)

$$\mathbf{H}(\mathbf{x}^*) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j} \right]_{(n \times n)} \quad (\text{b})$$

is positive semidefinite or positive definite at the point \mathbf{x}^* .

Second-order sufficiency condition. If the matrix $\mathbf{H}(\mathbf{x}^*)$ is positive definite at the stationary point \mathbf{x}^* , then \mathbf{x}^* is a local minimum point for the function $f(\mathbf{x})$.

Note that if $\mathbf{H}(\mathbf{x}^*)$ at the stationary point \mathbf{x}^* is indefinite, then \mathbf{x}^* is neither a local minimum nor a local maximum point because the second-order necessary condition is violated for both cases. Such stationary points are called *inflection points*. Also if $\mathbf{H}(\mathbf{x}^*)$ is at least positive semidefinite, then \mathbf{x}^* cannot be a local maximum since it violates the second-order necessary condition for a local maximum of $f(\mathbf{x})$. In other words a point cannot be a local minimum and local maximum simultaneously. The optimality conditions for a function of single variable and a function of several variables are summarized in Table 4-1.

Note also that these conditions involve derivatives of $f(\mathbf{x})$ and not the value of the function. If we *add a constant* to $f(\mathbf{x})$, the solution \mathbf{x}^* of the minimization problem remains unchanged, although the value of the cost function is altered. In a graph of $f(\mathbf{x})$ versus \mathbf{x} , adding a constant to $f(\mathbf{x})$ changes the origin of the coordinate system but leaves the shape of the surface unchanged. Similarly, if we multiply $f(\mathbf{x})$ by any positive constant the minimum point \mathbf{x}^* is unchanged but the value $f(\mathbf{x}^*)$ is altered. In a graph of $f(\mathbf{x})$ versus \mathbf{x} this is equi-

TABLE 4-1 Optimality Conditions for Unconstrained Problems

<i>Function of one variable minimize $f(x)$</i>	<i>Function of several variables minimize $f(\mathbf{x})$</i>
<i>First-order necessary condition:</i> $f' = 0$. Any point satisfying this condition is called a stationary point; it can be a local minimum, local maximum, or neither of the two (inflection point)	<i>First-order necessary condition:</i> $\nabla f = \mathbf{0}$. Any point satisfying this condition is called a stationary point; it can be a local minimum, local maximum, or neither of the two (inflection point)
<i>Second-order necessary condition</i> for a local minimum: $f'' \geq 0$	<i>Second-order necessary condition</i> for a local minimum: \mathbf{H} must be at least positive semidefinite
<i>Second-order necessary condition</i> for a local maximum: $f'' \leq 0$	<i>Second-order necessary condition</i> for a local maximum: \mathbf{H} must be at least negative semidefinite
<i>Second-order sufficient condition</i> for a local minimum: $f'' > 0$	<i>Second-order sufficient condition</i> for a local minimum: \mathbf{H} must be positive definite
<i>Second-order sufficient condition</i> for a local maximum: $f'' < 0$	<i>Second-order sufficient condition</i> for a local maximum: \mathbf{H} must be negative definite
<i>Higher-order necessary conditions</i> for a local minimum or local maximum: Calculate a higher ordered derivative that is not zero; all odd-ordered derivatives below this one must be zero	
<i>Higher-order sufficient condition</i> for a local minimum: Highest nonzero derivative must be even-ordered and positive	

valent to a uniform change of scale of the graph along the $f(\mathbf{x})$ axis, which again leaves the shape of the surface unaltered. Multiplying $f(\mathbf{x})$ by a negative constant changes the minimum at \mathbf{x}^* to a maximum. We may use this property to convert maximization problems to minimization problems by multiplying $f(\mathbf{x})$ by -1 . The effect of scaling and adding a constant to a function is shown in Example 4.19. In Examples 4.20 and 4.23, the local minima for a function are found using optimality conditions, while in Examples 4.21 and 4.22, the use of necessary conditions is explored.

EXAMPLE 4.19 Effects of Scaling or Adding a Constant to a Function

Discuss the effect of the preceding variations for the function $f(x) = x^2 - 2x + 2$.

Solution. Consider the graphs of Fig. 4-9. Figure 4-9(A) represents the function $f(x) = x^2 - 2x + 2$, which has a minimum at $x^* = 1$. Figures 4-9(B), (C), and (D) show the effect of adding a constant to the function [$f(x) + 1$], multiplying $f(x)$ by positive number [$2f(x)$], and multiplying it by a negative number [$-f(x)$]. In all cases, the stationary point remains unchanged.

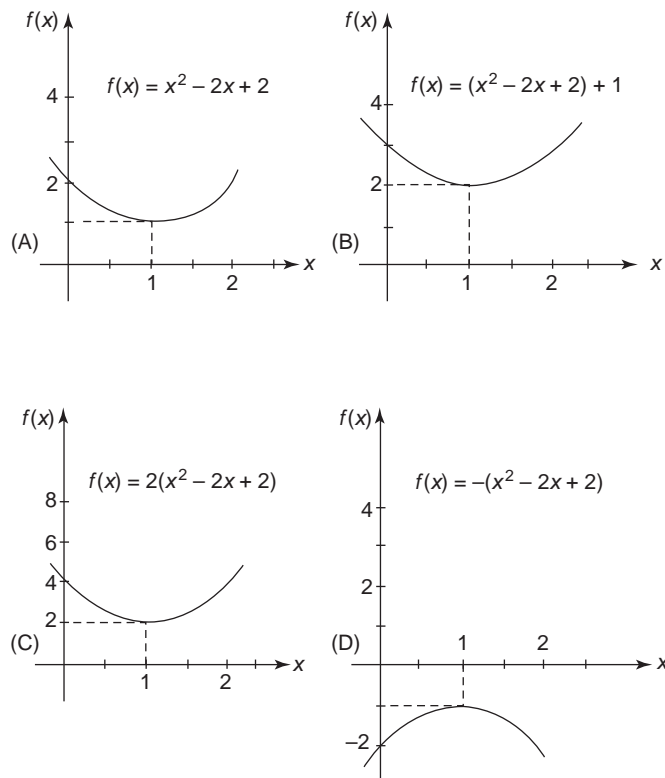


FIGURE 4-9 Graphs for Example 4.19. Effects of scaling or adding constant to a function. (A) Graph of $f(x) = x^2 - 2x + 2$. (B) Effect of addition of a constant to $f(x)$. (C) Effect of multiplying $f(x)$ by a positive constant. (D) Effect of multiplying $f(x)$ by -1 .

EXAMPLE 4.20 Local Minima for a Function of Two Variables Using Optimality Conditions

Find local minimum points for the function $f(\mathbf{x}) = x_1^2 + 2x_1x_2 + 2x_2^2 - 2x_1 + x_2 + 8$.

Solution. The necessary conditions for the problem give

$$\frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} (2x_1 + 2x_2 - 2) \\ (2x_1 + 4x_2 + 1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{a})$$

These equations are linear in variables x_1 and x_2 . Solving the equations simultaneously, we get the stationary point as $\mathbf{x}^* = (2.5, -1.5)$. To check if the stationary point is a local minimum, we evaluate \mathbf{H} at \mathbf{x}^* .

$$\mathbf{H}(2.5, -1.5) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \quad (\text{b})$$

By either of the tests of Theorems 4.2 and 4.3 or ($M_1 = 2 > 0$, $M_2 = 4 > 0$) or ($\lambda_1 = 5.236 > 0$, $\lambda_2 = 0.764 > 0$), \mathbf{H} is positive definite at the stationary point \mathbf{x}^* . Thus, it is a local minimum with $f(\mathbf{x}^*) = 4.75$. Figure 4-10 shows a few iso-cost curves for the function of this problem. It can be seen that the point $(2.5, -1.5)$ is the minimum for the function.

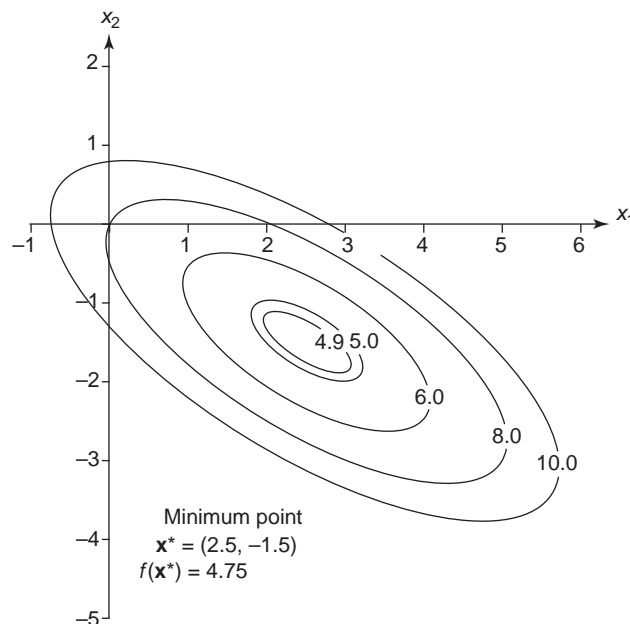


FIGURE 4-10 Isocost curves for the function of Example 4.20.

As noted earlier, the optimality conditions can also be used to check the optimality of a given point. To illustrate this, let us check the optimality of the point (1, 2). At this point, the gradient vector is calculated as (4, 11), which is not zero. Therefore the first-order necessary condition for a local minimum or a local maximum is violated and the point is not a stationary point.

EXAMPLE 4.21 Cylindrical Tank Design Using Necessary Conditions

In Section 2.8, a minimum cost cylindrical storage tank problem is formulated. The tank is closed at both ends and is required to have volume V . The radius R and height H are selected as design variables. It is desired to design the tank having minimum surface area. For the solution we may simplify the cost function as

$$\bar{f} = R^2 + RH \quad (a)$$

The volume constraint is an equality,

$$h = \pi R^2 H - V = 0 \quad (b)$$

This constraint cannot be satisfied if either R or H is zero. We may then neglect the non-negativity constraints on R and H if we agree to choose only the positive value for them. We may further use the equality constraint (b) to eliminate H from the cost function,

$$H = \frac{V}{\pi R^2} \quad (c)$$

Therefore, the cost function of Eq. (a) becomes

$$\bar{f} = R^2 + \frac{V}{\pi R} \quad (d)$$

This is an unconstrained problem in terms of R for which the necessary condition gives

$$\frac{d\bar{f}}{dR} = 2R - \frac{V}{\pi R^2} = 0 \quad (e)$$

The solution is

$$R^* = \left(\frac{V}{2\pi} \right)^{1/3} \quad (f)$$

Using Eq. (c), we obtain

$$H^* = \left(\frac{4V}{\pi} \right)^{1/3} \quad (g)$$

Using Eq. (e), the second derivative of \bar{f} with respect to R at the stationary point is

$$\frac{d^2\bar{f}}{dR^2} = \frac{2V}{\pi R^3} + 2 = 6 \quad (\text{h})$$

Since the second derivative is positive for all positive R , the solution in Eqs. (f) and (g) is a local minimum. Using Eqs. (a) or (d) the cost function at the optimum is given as

$$\bar{f}(R^*, H^*) = 3\left(\frac{V}{2\pi}\right)^{2/3} \quad (\text{i})$$

EXAMPLE 4.22 Numerical Solution of Necessary Conditions

Find stationary points for the following function and check sufficiency conditions for them:

$$f(x) = \frac{1}{3}x^2 + \cos x \quad (\text{a})$$

Solution. The function is plotted in Fig. 4-11. It can be seen that there are three stationary points: $x = 0$ (Point A), x between 1 and 2 (Point C), and x between -1 and -2 (Point B). The point $x = 0$ is a local maximum for the function and the other two are local minima.

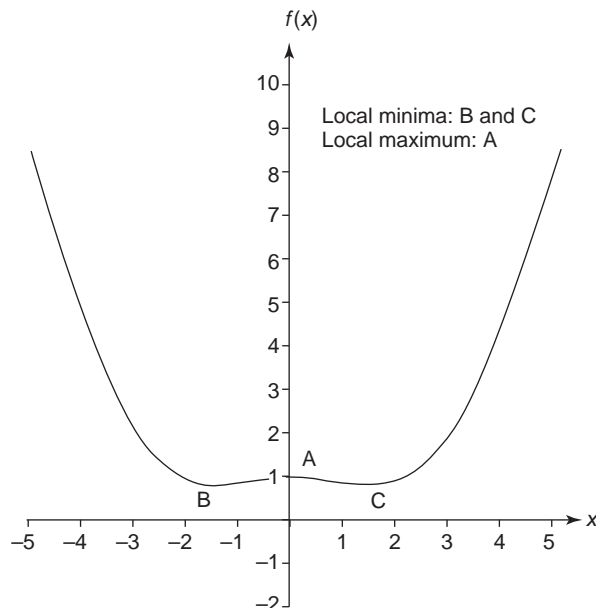


FIGURE 4-11 Graph of $f(x) = \frac{1}{3}x^2 + \cos x$ of Example 4.22.

The necessary condition is

$$f'(x) = \frac{2}{3}x - \sin x = 0 \quad (b)$$

It can be seen that $x = 0$ satisfies Eq. (b), so it is a stationary point. We must find other roots of Eq. (b). Finding an analytical solution for the equation is difficult, so we must use numerical methods. We can either plot $f'(x)$ versus x and locate the point where $f'(x) = 0$, or use a numerical method for solving nonlinear equations. A numerical method for solving such an equation known as the *Newton-Raphson method* is given in Appendix C. By either of the two methods, we find that $x^* = 1.496$ and -1.496 satisfy $f'(x) = 0$ in Eq. (b). Therefore, these are additional stationary points. To determine whether they are local minimum, maximum, or inflection points, we must determine f'' at the stationary points and use sufficient conditions of Theorem 4.4. Since $f'' = \frac{2}{3} - \cos x$, we have

1. $x^* = 0$; $f'' = -\frac{1}{3} < 0$, so this is a local maximum with $f(0) = 1$.
2. $x^* = 1.496$; $f'' = 0.592 > 0$, so this is a local minimum with $f(1.496) = 0.821$.
3. $x^* = -1.496$; $f'' = 0.592 > 0$, so this is a local minimum with $f(-1.496) = 0.821$.

These results agree with the graphical solutions observed in Fig. 4-11. Note that $x^* = 1.496$ and -1.496 are actually global minimum points for the function although the function is unbounded and the feasible set is not closed. Note also that there is no global maximum point for the function since the function is unbounded and x is allowed to have any value.

EXAMPLE 4.23 Local Minima for a Function of Two Variables Using Optimality Conditions

Find a local minimum point for the function

$$f(\mathbf{x}) = x_1 + \frac{(4 \times 10^6)}{x_1 x_2} + 250x_2$$

Solution. The necessary conditions for optimality are

$$\frac{\partial f}{\partial x_1} = 0; \quad 1 - \frac{(4 \times 10^6)}{x_1^2 x_2} = 0 \quad (a)$$

$$\frac{\partial f}{\partial x_2} = 0; \quad 250 - \frac{(4 \times 10^6)}{x_1 x_2^2} = 0 \quad (b)$$

Equations (a) and (b) give

$$x_1^2 x_2 - (4 \times 10^6) = 0; \quad 250x_1 x_2^2 - (4 \times 10^6) = 0 \quad (c)$$

These equations give

$$x_1^2 x_2 = 250 x_1 x_2^2, \quad \text{or} \quad x_1 x_2 (x_1 - 250 x_2) = 0 \quad (d)$$

Since neither x_1 nor x_2 can be zero (the function has singularity at $x_1 = 0$ or $x_2 = 0$), the preceding equation gives $x_1 = 250 x_2$. Substituting this into Eq. (b), we obtain $x_2 = 4$. Therefore, $x_1^* = 1000$, and $x_2^* = 4$ is a stationary point for the function $f(\mathbf{x})$. Using Eqs. (a) and (b), the Hessian matrix for $f(\mathbf{x})$ at the point \mathbf{x}^* is given as

$$\mathbf{H} = \frac{(4 \times 10^6)}{x_1^2 x_2^2} \begin{bmatrix} \frac{2x_2}{x_1} & 1 \\ 1 & \frac{2x_1}{x_2} \end{bmatrix}; \quad \mathbf{H}(1000, 4) = \frac{(4 \times 10^6)}{(4000)^2} \begin{bmatrix} 0.008 & 1 \\ 1 & 500 \end{bmatrix} \quad (e)$$

Eigenvalues of the above Hessian (without the constant of $\frac{1}{4}$) are: $\lambda_1 = 0.006$ and $\lambda_2 = 500.002$. Since both eigenvalues are positive, the Hessian of $f(\mathbf{x})$ at the point \mathbf{x}^* is positive definite. Therefore, $\mathbf{x}^* = (1000, 4)$ is a local minimum point with $f(\mathbf{x}^*) = 3000$. Figure 4-12 shows some isocost curves for the function of this problem. It can be seen that $x_1 = 1000$ and $x_2 = 4$ is the minimum point. (Note that the horizontal and vertical scales are quite different in Fig. 4-12; this is done to obtain reasonable isocost curves.)

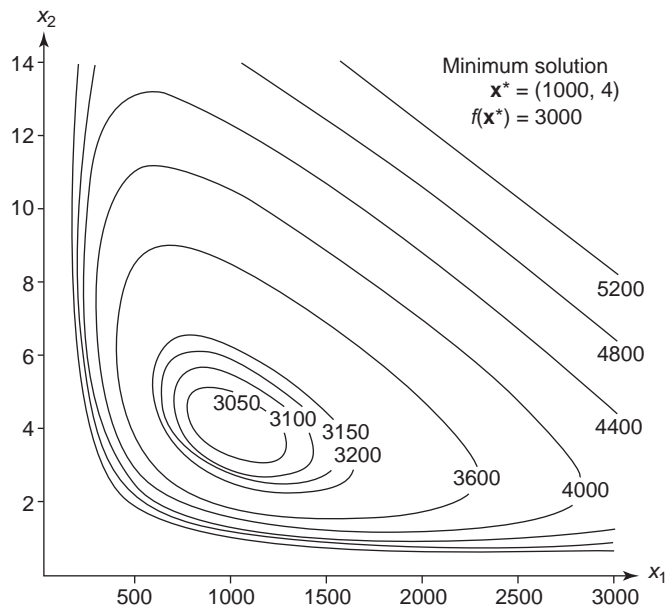


FIGURE 4-12 Isocost curves for the function of Example 4.23.

4.3.4 Roots of Nonlinear Equations Using Excel

Excel is a spreadsheet program that has many useful capabilities for engineering calculations. In particular it can be used to solve equations and optimization problems. Online help is available to use the program. When the program is invoked, it opens what is called the “work-

	A	B	C	D
1		Roots of $2x/3 - \sin x = 0$		
2				
3	Variable:	x	1	
4	Equation:	$2x/3 - \sin x$	-0.17480432	
5				

FIGURE 4-13 Excel worksheet for finding roots of $\frac{2}{3}x - \sin x = 0$.

book.” A workbook contains several pages called “worksheets.” These worksheets may be used to store related data and information. Each worksheet is divided into cells that are referenced by their location, the column-row number. All the information, data, and its manipulation must be organized in terms of cells. Cells may contain raw data, formulas, and references to other cells.

“Solver” is the tool available in Excel to solve a nonlinear equation, a system of linear/nonlinear equations, and optimization problems. Here, we shall use Solver to find the roots of the nonlinear equation $\frac{2}{3}x - \sin x = 0$. This equation was obtained as a necessary condition for the minimization problem of Example 4.22.

Solver is invoked through the “Tools” menu. If it is not visible under Tools, it is not installed yet. To install it, use the “Add-in” command under Tools menu.

We need to prepare a worksheet that defines the problem. The worksheet can be prepared in many different ways, and Fig. 4-13 shows one such way.

We define cell C3 as x , the solution variable. To name a cell, use the Insert/Name/Define command and click the Add button to define names for cells. Defining meaningful names for cells is useful because they can be referred to by their names rather than their cell numbers. Cells contain the following information:

Cell A3: indicates that Row 3 is associated with variable x .

Cell A4: indicates that Row 4 is associated with the equation.

Cell B3: variable name that will appear later in the “Answer Report.”

Cell C3: starting value for the variable in cell B3, named x ; updated later by Solver to a final value.

Cell B4: equation whose roots need to be found; will appear later in the “Answer Report.”

Cell C4: contains expression for the equation as “ $=2*x/3 - \sin(x)$ ”; currently it displays the value of the expression for $x = 1$ in cell C3.

Whenever the value in cell C3 is changed, cell C4 gets updated automatically. Now Solver is invoked from the Tools menu to define the target cell and the cell to be varied. Figure 4-14 shows the dialog box for the Solver. We set the target cell as C4 because that contains the equation whose roots need to be determined. We set its value to zero in the dialog box because when the root is found its value should be zero. (Solver uses the “\$” symbol to identify cells; for example, $\$C\4 refers to cell C4. Use of \$ in a cell reference is convenient when the “copy formula” capability is used. With the “copy formula” command, the reference to a cell with the \$ symbol is not changed.)

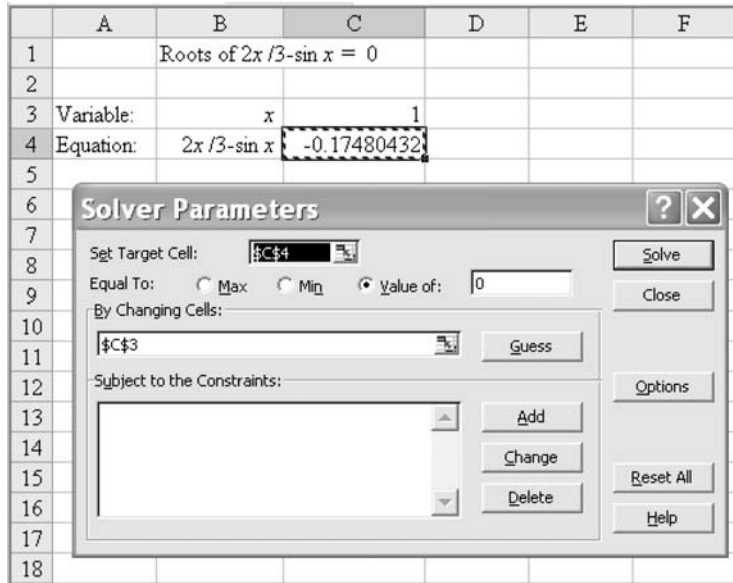


FIGURE 4-14 Solver dialog box to set parameters for the problem.

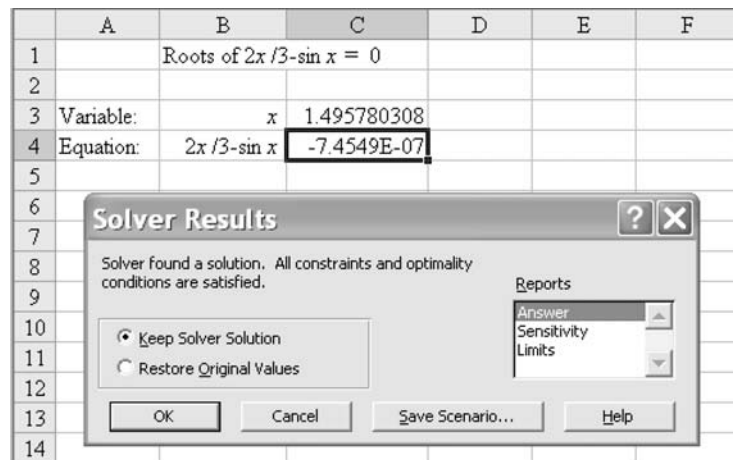


FIGURE 4-15 Solver results dialog box and the final worksheet.

Next we define the cell whose value needs to be treated as a variable by inserting C3 in the box “By Changing Cells.” Then by pressing the “Options” button we can reset some of the parameters related to the solution procedure, if desired. Otherwise, we press the “Solve” button to find a root starting from $x = 1$. When Solver is finished, it produces the three reports, Answer, Sensitivity, and Limits, seen in the “Solver Results” dialog box in Fig. 4-15. We select only the “Answer Report” because that contains the relevant information. When “OK” is pressed with the “Answer” option highlighted under “Reports,” Solver produces another worksheet that contains the final results, as shown in Fig. 4-16. Other roots for the equation can be found by changing the starting value for x in cell C3. If one starting value does not work, another value should be tried. Using this procedure, three roots of the equation are found as 0, 1.496, and -1.496 , as before.

	A	B	C	D	E	
1						
2	Target Cell (Value Of)					
3	Cell		Name		Original Value	Final Value
4	\$C\$4		2x /3-sin x		-0.174804318	-7.45491E-07
5						
6	Adjustable Cells					
7	Cell		Name		Original Value	Final Value
8	\$C\$3		x		1	1.495780308
9						
10	Constraints					
11	NONE					

FIGURE 4-16 Solver Answer Report for roots of $\frac{2}{3}x - \sin x = 0$.

4.4 Constrained Optimum Design Problems

We have seen in Chapter 2 that most design problems include *constraints on variables and on performance* of the system. This section describes concepts related to constrained optimization problems. The necessary conditions for an equality constrained problem are discussed first. These conditions are contained in the *Lagrange Multiplier theorem* generally discussed in textbooks on calculus. Then, the necessary conditions for the general constrained problem are obtained as an extension of the Lagrange Multiplier theorem. These are known as *Karush-Kuhn-Tucker (KKT) necessary conditions*. The necessary conditions for optimality are explained and illustrated with examples. All optimum designs must satisfy these conditions. The standard design optimization model introduced in Section 2.11 is restated in Table 4-2. The inequality constraints of Eq. (4.39) will be initially ignored to discuss the Lagrange theorem given in calculus books. The theorem will then be extended for inequality constraints.

TABLE 4-2 General Design Optimization Model

Design variable vector:	$\mathbf{x} = (x_1, x_2, \dots, x_n)$	
Cost function:	$f(\mathbf{x}) = f(x_1, x_2, \dots, x_n)$	(4.37)
Equality constraints:	$h_i(\mathbf{x}) = 0; \quad i = 1 \text{ to } p$	(4.38)
Inequality constraints:	$g_i(\mathbf{x}) \leq 0; \quad i = 1 \text{ to } m$	(4.39)

4.4.1 Role of Constraints

Based on the discussion of unconstrained optimization problems, one might conclude that only the nature of the cost function $f(\mathbf{x})$ for the constrained problems will determine the location of the minimum point. This, however, is not true because the constraint functions also play a prominent role in determining the optimum solution. Examples 4.24 and 4.25 illustrate these situations.

EXAMPLE 4.24 Constrained Optimum Point

Minimize $f(\mathbf{x}) = (x_1 - 1.5)^2 + (x_2 - 1.5)^2$ subject to the constraints written in the standard form

$$g_1(\mathbf{x}) = x_1 + x_2 - 2 \leq 0 \quad (\text{a})$$

$$g_2(\mathbf{x}) = -x_1 \leq 0; \quad g_3(\mathbf{x}) = -x_2 \leq 0 \quad (\text{b})$$

Solution. The feasible set S for the problem is a triangular region shown in Fig. 4-17. If constraints are ignored, $f(\mathbf{x})$ has a minimum at the point $(1.5, 1.5)$ which violates the constraint g_1 and therefore is an infeasible point. Note that contours of $f(\mathbf{x})$ are circles. They increase in diameter as $f(\mathbf{x})$ increases. It is clear that the minimum value for $f(\mathbf{x})$ corresponds to a circle with the smallest radius intersecting the feasible set. This is the point $(1, 1)$ at which $f(\mathbf{x}) = 0.5$. The point is on the boundary of the feasible region. Thus, the location of the optimum point is governed by the constraints for this problem as well as cost function contours.

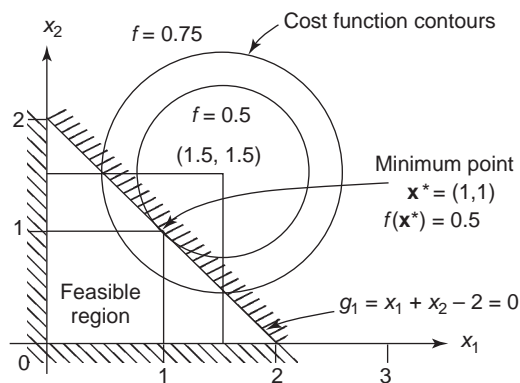


FIGURE 4-17 Graphical representation for Example 4.24. Constrained optimum point.

EXAMPLE 4.25 Unconstrained Optimum Point for a Constrained Problem

Minimize $f(\mathbf{x}) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2$ subject to the same constraints as in Example 4.24.

Solution. The feasible set S is the same as in Example 4.24. The cost function, however, has been modified. If constraints are ignored, $f(\mathbf{x})$ has a minimum at $(0.5, 0.5)$. Since the point also satisfies all the constraints, it is the optimum solution. The solution for this problem therefore occurs in the interior of the feasible region and the constraints play no role in its location.

Note that a *solution to a constrained optimization problem may not exist*. This can happen if we over-constrain the system. The requirements can be conflicting such that it is impossible to build a system to satisfy them. In such a case we must re-examine the problem formulation and relax constraints. Example 4.26 illustrates the situation.

EXAMPLE 4.26 Infeasible Problem

Minimize $f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 2)^2$ subject to the constraints written in the standard form

$$g_1(\mathbf{x}) = x_1 + x_2 - 2 \leq 0 \quad (\text{a})$$

$$g_2(\mathbf{x}) = -x_1 + x_2 + 3 \leq 0 \quad (\text{b})$$

$$g_3(\mathbf{x}) = -x_1 \leq 0; \quad g_4(\mathbf{x}) = -x_2 \leq 0 \quad (\text{c})$$

Solution. Figure 4-18 shows a plot of the constraints for the problem. It can be seen that there is no design satisfying all the constraints. The feasible set S for the problem is empty and there is no solution (i.e., no feasible design).

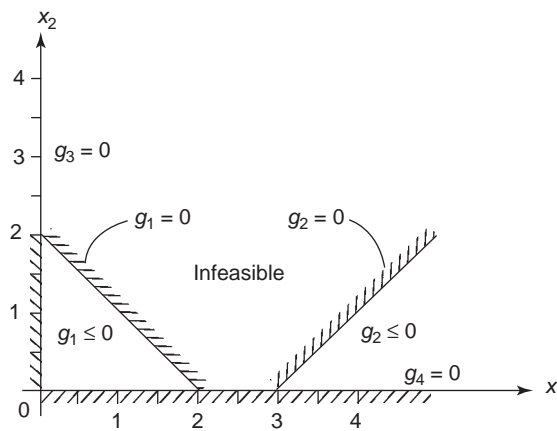


FIGURE 4-18 Plot of constraints for Example 4.26. Infeasible problem.

4.4.2 Necessary Conditions: Equality Constraints

We first discuss the necessary conditions for constrained minimization problems with only equality constraints. Just as for the unconstrained case, solutions of these conditions give candidate minimum points. The sufficient conditions—discussed in Chapter 5—can be used to determine if a candidate point is indeed a local minimum. Extensions to include inequalities are described in the next subsection.

**Regular Point* Before discussing the necessary conditions, we define a *regular point* of the feasible set (design space). Consider the constrained optimization problem of minimizing $f(\mathbf{x})$ subject to the constraints $h_i(\mathbf{x}) = 0$, $i = 1$ to p . A point \mathbf{x}^* satisfying the constraints $h(\mathbf{x}^*) = 0$ is said to be a *regular point* of the feasible set if $f(\mathbf{x}^*)$ 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 (Appendix B). When inequality constraints are also included in the problem definition, then for a point to be regular, gradients of active inequalities must be also linearly independent.

Lagrange Multipliers and Necessary Conditions It turns out that a scalar multiplier is associated with each constraint, called the *Lagrange multiplier*. These multipliers play a prominent role in optimization theory as well as numerical methods. The multipliers have a geometrical as well as physical meaning. Their values depend on the form of the cost and constraint functions. If these functions change, the values of the Lagrange multipliers also change. We shall discuss this aspect later in Section 4.5. To introduce the idea of Lagrange multipliers, we consider Example 4.27, which minimizes a cost function of two variables with one equality constraint.

EXAMPLE 4.27 Introduction of Lagrange Multipliers and Their Geometrical Meaning

Minimize

$$f(x_1, x_2) = (x_1 - 1.5)^2 + (x_2 - 1.5)^2 \quad (\text{a})$$

subject to an equality constraint

$$h(x_1, x_2) = x_1 + x_2 - 2 = 0 \quad (\text{b})$$

Solution. The problem has two variables and can be solved easily by the graphical procedure. Figure 4-19 shows a graphical representation for the problem. The straight line A–B represents the equality constraint and the feasible region for the problem. Therefore, the optimum solution must lie on the line A–B. The cost function is an equation of a circle with its center at point (1.5, 1.5). The isocost curves, having values of 0.5 and 0.75, are shown in the figure. It can be seen that point C, having coordinates (1, 1), gives the optimum solution for the problem. The cost function contour of value 0.5 just touches the line A–B, so this is the minimum value for the cost function.

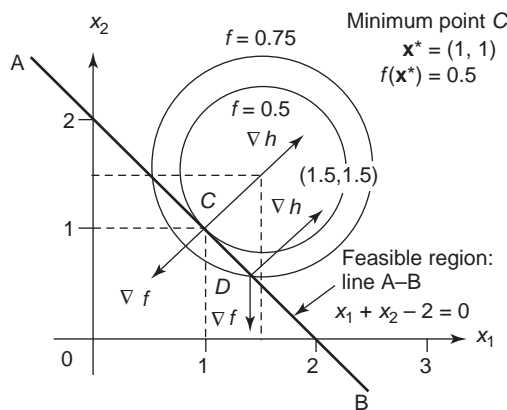


FIGURE 4-19 Graphical solution for Example 4.27. Geometrical interpretation of necessary conditions.

Introduction of Lagrange multipliers. Now let us see what mathematical conditions are satisfied at the minimum point C. Let the optimum point be represented as (x_1^*, x_2^*) . To derive the conditions and to introduce the Lagrange multiplier, we first assume that the equality constraint can be used to solve for one variable in terms of the other (at least symbolically), i.e., assume that we can write

$$x_2 = \phi(x_1) \quad (c)$$

where ϕ is an appropriate function of x_1 . In many problems, it may not be possible to write explicitly the function $\phi(x_1)$, but for derivation purposes, we assume its existence. It will be seen later that the explicit form of this function is not needed. For the present example, $\phi(x_1)$ from Eq. (b) is given as

$$x_2 = \phi(x_1) = -x_1 + 2 \quad (d)$$

Substituting Eq. (c) into Eq. (a), we eliminate x_2 from the cost function and obtain the unconstrained minimization problem in terms of x_1 only:

$$\text{minimize } f(x_1, \phi(x_1)) \quad (e)$$

For the present example, substituting Eq. (d) into Eq. (a), we eliminate x_2 and obtain the minimization problem in terms of x_1 alone:

$$f(x_1) = (x_1 - 1.5)^2 + (-x_1 + 2 - 1.5)^2$$

The necessary condition $df/dx_1 = 0$ gives $x_1^* = 1$. Then Eq. (d) gives $x_2^* = 1$ and the cost function at the point $(1, 1)$ is 0.5. It can be checked that the sufficiency condition $d^2f/dx_1^2 > 0$ is also satisfied, and so the point is indeed a local minimum as seen in Fig. 4-19.

If we assume that the explicit form of the function $\phi(x_1)$ cannot be obtained (which is generally the case), then some alternate procedure must be developed to obtain the optimum solution. We shall derive such a procedure and see that the Lagrange multiplier for the constraint gets introduced naturally in the process. Using the chain rule of differentiation, we write the necessary condition $df/dx_1 = 0$ for the problem defined in Eq. (e) as

$$\frac{df(x_1, x_2)}{dx_1} = \frac{\partial f(x_1, x_2)}{\partial x_1} + \frac{\partial f(x_1, x_2)}{\partial x_2} \frac{dx_2}{dx_1} = 0$$

Substituting Eq. (c), the preceding equation can be written at the optimum point (x_1^*, x_2^*) as

$$\frac{\partial f(x_1^*, x_2^*)}{\partial x_1} + \frac{\partial f(x_1^*, x_2^*)}{\partial x_2} \frac{d\phi}{dx_1} = 0 \quad (f)$$

Since ϕ is not known, we need to eliminate $d\phi/dx_1 = 0$ from Eq. (f). To accomplish this, we differentiate the constraint equation $h(x_1, x_2) = 0$ at the point (x_1^*, x_2^*) as

$$\frac{dh(x_1^*, x_2^*)}{dx_1} = \frac{\partial h(x_1^*, x_2^*)}{\partial x_1} + \frac{\partial h(x_1^*, x_2^*)}{\partial x_2} \frac{d\phi}{dx_1} = 0$$

Or, solving for $d\phi/dx_1$, we obtain (assuming $\partial h/\partial x_2 \neq 0$)

$$\frac{d\phi}{dx_1} = -\frac{\partial h(x_1^*, x_2^*)/\partial x_1}{\partial h(x_1^*, x_2^*)/\partial x_2} \quad (g)$$

Now substituting for $d\phi/dx_1$ from Eq. (g) into Eq. (f), we obtain

$$\frac{\partial f(x_1^*, x_2^*)}{\partial x_1} - \frac{\partial f(x_1^*, x_2^*)}{\partial x_2} \left(\frac{\partial h(x_1^*, x_2^*)/\partial x_1}{\partial h(x_1^*, x_2^*)/\partial x_2} \right) = 0 \quad (h)$$

If we define a quantity v as

$$v = -\frac{\partial f(x_1^*, x_2^*)/\partial x_2}{\partial h(x_1^*, x_2^*)/\partial x_2} \quad (i)$$

and substitute it into Eq. (h), we obtain

$$\frac{\partial f(x_1^*, x_2^*)}{\partial x_1} + v \frac{\partial h(x_1^*, x_2^*)}{\partial x_1} = 0 \quad (j)$$

Also, rearranging Eq. (i) that defines v , we obtain

$$\frac{\partial f(x_1^*, x_2^*)}{\partial x_2} + v \frac{\partial h(x_1^*, x_2^*)}{\partial x_2} = 0 \quad (k)$$

Equations (j) and (k) along with the equality constraint $h(x_1, x_2) = 0$ are the *necessary conditions of optimality for the problem*. Any point that violates these conditions cannot be a minimum point for the problem. The scalar quantity v defined in Eq. (i) is called the *Lagrange multiplier*. If the minimum point is known, Eq. (i) can be used to calculate its value. For the present example, $\partial f(1, 1)/\partial x_2 = -1$ and $\partial h(1, 1)/\partial x_2 = 1$; therefore, Eq. (i) gives $v^* = 1$ as the Lagrange multiplier at the optimum point.

Recall that the necessary conditions can be used to solve for the candidate minimum points, i.e., Eqs. (j), (k), and $h(x_1, x_2) = 0$ can be used to solve for x_1 , x_2 , and v . For the present example, these equations give

$$2(x_1 - 1.5) + v = 0; \quad 2(x_2 - 1.5) + v = 0; \quad x_1 + x_2 - 2 = 0 \quad (l)$$

Solution of the preceding equations is indeed, $x_1^* = 1$, $x_2^* = 1$, and $v^* = 1$.

Geometrical meaning of Lagrange multipliers. It is customary to use what is known as the *Lagrange function* in writing the necessary conditions. The Lagrange function is denoted as L and defined using cost and constraint functions as

$$L(x_1, x_2, v) = f(x_1, x_2) + vh(x_1, x_2) \quad (m)$$

It is seen that the necessary conditions of Eqs. (j) and (k) are given in terms of L as

$$\frac{\partial L(x_1^*, x_2^*)}{\partial x_1} = 0, \quad \frac{\partial L(x_1^*, x_2^*)}{\partial x_2} = 0 \quad (n)$$

Or, in the vector notation we see that the gradient of L is zero at the candidate minimum point, i.e., $\nabla L(x_1^*, x_2^*) = \mathbf{0}$. Writing this condition, using Eq. (m), or writing Eqs. (j) and (k) in the vector form, we obtain

$$\nabla f(\mathbf{x}^*) + v\nabla h(\mathbf{x}^*) = \mathbf{0} \quad (\text{o})$$

where gradients of the cost and constraint functions are given as

$$\nabla f(\mathbf{x}^*) = \begin{bmatrix} \frac{\partial f(x_1^*, x_2^*)}{\partial x_1} \\ \frac{\partial f(x_1^*, x_2^*)}{\partial x_2} \end{bmatrix}, \quad \nabla h = \begin{bmatrix} \frac{\partial h(x_1^*, x_2^*)}{\partial x_1} \\ \frac{\partial h(x_1^*, x_2^*)}{\partial x_2} \end{bmatrix} \quad (\text{p})$$

Equation (o) can be rearranged as

$$\nabla f(\mathbf{x}^*) = -v\nabla h(\mathbf{x}^*) \quad (\text{q})$$

The preceding equation brings out the geometrical meaning of the necessary conditions. It shows that at the candidate minimum point, gradients of the cost and constraint functions are along the same line and proportional to each other, and the Lagrange multiplier v is the proportionality constant.

For the present example, the gradients of cost and constraint functions at the candidate optimum point are given as

$$\nabla f(1, 1) = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \nabla h(1, 1) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (\text{r})$$

These vectors are shown at point C in Fig. 4-19. Note that they are along the same line. For any feasible point, say (0.4, 1.6), on line A–B other than the candidate minimum, the gradients of cost and constraint functions will not be along the same line, as seen in the following:

$$\nabla f(0.4, 1.6) = \begin{bmatrix} -2.2 \\ 0.2 \end{bmatrix}, \quad \nabla h(0.4, 1.6) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (\text{s})$$

As another example, point D in Fig. 4-19 is not a candidate minimum, since gradients of cost and constraint functions are not along the same line. Also, the cost function has higher value at these points compared to the one at the minimum point, i.e., we can move away from point D toward point C and reduce the cost function.

It is interesting to note that the equality constraint can be multiplied by -1 without affecting the minimum point, i.e., the constraint can be written as $-x_1 - x_2 + 2 = 0$. The minimum solution is still the same $x_1^* = 1, x_2^* = 1$ and $f(\mathbf{x}^*) = 0.5$; however, the sign of the Lagrange multiplier is reversed, i.e., $v^* = -1$. *This shows that the Lagrange multiplier for the equality constraint is free in sign*, i.e., the sign is determined by the form of the constraint function.

It is also interesting to note that any small move from point C in the feasible region (i.e., along the line A–B) increases the cost function value, and *any further reduction to the cost function is accompanied by violation of the constraint*. Thus, point C satisfies the sufficiency condition for a local minimum point because it has the smallest value in a neighborhood of point C [note that we have used the definition of a local minimum given in Eq. (4.1), Section 4.1.1]. Thus, it is indeed a local minimum point.

The concept of Lagrange multipliers is quite general. It is encountered in many engineering applications other than the optimum design. *The Lagrange multiplier for a constraint can be interpreted as the force required to impose the constraint.* We shall discuss a physical meaning of the Lagrange multipliers in Section 4.5. The idea of a Lagrange multiplier for an equality constraint can be generalized to many equality constraints. It can also be extended for inequality constraints. We shall first discuss the necessary conditions with multiple equality constraints in Theorem 4.5 and then describe in the next section their extensions to include the inequality constraints.

Theorem 4.5 Lagrange Multiplier Theorem Consider the problem of minimizing $f(\mathbf{x})$ subject to the equality constraints $h_i(\mathbf{x}) = 0, i = 1$ to p . Let \mathbf{x}^* be a regular point that is a local minimum for the problem. Then there exist unique Lagrange multipliers $v_j^*, j = 1$ to p such that

$$\frac{\partial f(\mathbf{x}^*)}{\partial x_i} + \sum_{j=1}^p v_j^* \frac{\partial h_j(\mathbf{x}^*)}{\partial x_i} = 0; \quad i = 1 \text{ to } n \quad (4.40)$$

$$h_j(\mathbf{x}^*) = 0; \quad j = 1 \text{ to } p$$

It is convenient to write these conditions in terms of a *Lagrange function* defined as

$$L(\mathbf{x}, \mathbf{v}) = f(\mathbf{x}) + \sum_{j=1}^p v_j h_j(\mathbf{x}) = f(\mathbf{x}) + \mathbf{v}^T \mathbf{h}(\mathbf{x}) \quad (4.41)$$

Then Eq. (4.40) becomes

$$\nabla L(\mathbf{x}^*, \mathbf{v}^*) = \mathbf{0}, \quad \text{or} \quad \frac{\partial L(\mathbf{x}^*, \mathbf{v}^*)}{\partial x_i} = 0; \quad i = 1 \text{ to } n \quad (4.42)$$

Differentiating $L(\mathbf{x}, \mathbf{v})$ with respect to v_j we can recover the equality constraints as

$$\frac{\partial L(\mathbf{x}^*, \mathbf{v}^*)}{\partial v_j} = 0 \Rightarrow h_j(\mathbf{x}^*) = 0; \quad j = 1 \text{ to } p \quad (4.43)$$

The gradient conditions of Eqs. (4.42) and (4.43) show that the Lagrange function is stationary with respect to both \mathbf{x} and \mathbf{v} . Therefore, it may be treated as an unconstrained function in the variables \mathbf{x} and \mathbf{v} to determine the stationary points. *Note that any point that does not satisfy conditions of the theorem 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 given in Chapter 5 can distinguish between the minimum, maximum, and inflection points.

The n variables \mathbf{x} and the p multipliers \mathbf{v} are the unknowns, and the necessary conditions of Eqs. (4.42) and (4.43) provide enough equations to solve for them. Note also that the Lagrange multipliers v_i are free in sign, i.e., they can be positive, negative, or zero. This is in contrast to the Lagrange multipliers for the inequality constraints, which we shall see later are required to be nonnegative.

The gradient condition of Eq. (4.40) can be rearranged as

$$\frac{\partial f(\mathbf{x}^*)}{\partial x_i} = - \sum_{j=1}^p v_j^* \frac{\partial h_j(\mathbf{x}^*)}{\partial x_i}; \quad i = 1 \text{ to } n$$

This form shows that the *gradient of the cost function is a linear combination of the gradients of the constraints at the candidate minimum point*. The Lagrange multipliers v_j^* act as the scalars of the linear combination. This linear combination interpretation of the necessary conditions is a generalization of the concept discussed in Example 4.27 for one constraint: “at the candidate minimum point gradients of the cost and constraint functions are along the same line.” Example 4.28 illustrates the necessary conditions for an equality constrained problem.

EXAMPLE 4.28 Cylindrical Tank Design—Use of Lagrange Multipliers for a Equality Constrained Problem

We will re-solve the cylindrical storage tank problem (Example 4.21) using the Lagrange multiplier approach. The problem is to find radius R and length H of the cylinder to minimize $f = R^2 + RH$ subject to $h = \pi R^2 H - V = 0$.

Solution. The Lagrange function L for the problem is given as

$$L = R^2 + RH + v(\pi R^2 H - V) \quad (a)$$

The necessary conditions of the Lagrange Multiplier Theorem 4.5 give

$$\frac{\partial L}{\partial R} = 2R + H + 2\pi v R H = 0 \quad (b)$$

$$\frac{\partial L}{\partial H} = R + \pi v R^2 = 0 \quad (c)$$

$$\frac{\partial L}{\partial v} = \pi R^2 H - V = 0 \quad (d)$$

These are three equations in three unknown v , R , and H . Note that they are nonlinear. However, they can be easily solved by the elimination process giving the solution of the necessary conditions as

$$R^* = \left(\frac{V}{2\pi}\right)^{1/3}; \quad H^* = \left(\frac{4V}{\pi}\right)^{1/3}; \quad v^* = -\frac{1}{\pi R} = -\left(\frac{2}{\pi^2 V}\right)^{1/3} \quad (e)$$

This is the same solution as obtained for Example 4.21, treating it as an unconstrained problem. It can be also verified that the gradients of the cost and constraint functions are along the same line at the optimum point. Note that this problem has only one equality constraint. Therefore the question of linear dependence of the gradients of active constraints does not arise; i.e., the regularity condition for the solution point is satisfied.

Often, the necessary conditions of the Lagrange Multiplier Theorem lead to a nonlinear set of equations that cannot be solved analytically. In such cases, we must use a numerical algorithm such as the Newton-Raphson method (Appendix C) to solve for their roots and the candidate minimum points. Several commercial software packages, such as Excel, MATLAB, and Mathematica, are also available to find roots of nonlinear equations.

4.4.3 Necessary Conditions: Inequality Constraints—Karush-Kuhn-Tucker (KKT) Conditions

The design problem formulations in Chapter 2 often included inequality constraints of the form $g_i(\mathbf{x}) \leq 0$. We can transform an inequality constraint to equality by adding a new variable to it, called the *slack variable*. Since the constraint is of the form “ \leq ”, its value is either negative or zero. 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 $g_i(\mathbf{x}) + s_i = 0$, where $s_i \geq 0$ is a slack variable. The variables s_i are treated as unknowns of the design problem along with the original 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 inequality is called an *active (tight) constraint*, i.e., there is no “slack” in the constraint. For any $s_i > 0$, the corresponding constraint is a strict inequality. It is called an *inactive constraint*, and has slack given by s_i .

Note that with the preceding procedure, we must introduce one additional variable s_i and an additional constraint $s_i \geq 0$ to treat each inequality constraint. This increases the dimension of the design problem. The constraint $s_i \geq 0$ can be avoided if we use s_i^2 as the slack variable instead of just s_i . Therefore, the inequality $g_i \leq 0$ is converted to equality as

$$g_i + s_i^2 = 0 \quad (4.44)$$

where s_i can have any real value. This form can be used in the Lagrange Multiplier Theorem to treat inequality constraints and to derive the corresponding necessary conditions. The m new equations needed for determining the slack variables are obtained by requiring the Lagrangian L to be stationary with respect to the slack variables as well ($\partial L / \partial \mathbf{s} = \mathbf{0}$).

Note that once a design point is specified, Eq. (4.44) can be used to calculate the slack variable s_i^2 . If the constraint is satisfied at the point (i.e., $g_i \leq 0$), then $s_i^2 \geq 0$. If it is violated, then s_i^2 is negative which is not acceptable, i.e., the point is not a candidate minimum point. There is an *additional necessary condition* for the Lagrange multipliers of “ \leq type” constraints given as

$$u_j^* \geq 0; \quad j = 1 \text{ to } m \quad (4.45)$$

where u_j^* is the Lagrange multiplier for the j th inequality constraint. Thus, *the Lagrange multiplier for each “ \leq ” inequality constraint must be nonnegative*. If the constraint is inactive at the optimum, its associated Lagrange multiplier is zero. If it is active ($g_i = 0$), then the associated multiplier must be nonnegative. We will explain the condition of Eq. (4.45) from a physical point of view in Section 4.5. Example 4.29 illustrates the use of necessary conditions in an inequality constrained problem.

EXAMPLE 4.29 Inequality Constrained Problem—Use of Necessary Conditions

We will re-solve Example 4.27 by treating the constraint as an inequality. The problem is to minimize $f(x_1, x_2) = (x_1 - 1.5)^2 + (x_2 - 1.5)^2$ subject to an inequality written in the standard form as $g(\mathbf{x}) = x_1 + x_2 - 2 \leq 0$.

Solution. The graphical representation for the problem remains the same as in Fig. 4-19 for Example 4.27, except that the feasible region is enlarged; it is line A-B and the region below it. The minimum point for the problem is same as before, i.e., $x_1^* = 1$, $x_2^* = 1$, $f(\mathbf{x}^*) = 0.5$. Introducing a slack variable s^2 for the inequality, the Lagrange function of Eq. (4.41) for the problem is defined as

$$L = (x_1 - 1.5)^2 + (x_2 - 1.5)^2 + u(x_1 + x_2 - 2 + s^2) \quad (\text{a})$$

where u is the Lagrange multiplier for the inequality constraint. The necessary conditions of the Lagrange Theorem give (treating x_1 , x_2 , u , as unknowns)

$$\frac{\partial L}{\partial x_1} = 2(x_1 - 1.5) + u = 0 \quad (\text{b})$$

$$\frac{\partial L}{\partial x_2} = 2(x_2 - 1.5) + u = 0 \quad (\text{c})$$

$$\frac{\partial L}{\partial u} = x_1 + x_2 - 2 + s^2 = 0 \quad (\text{d})$$

$$\frac{\partial L}{\partial s} = 2us = 0 \quad (\text{e})$$

These are four equations in four unknowns x_1 , x_2 , u , and s . The equations must be solved simultaneously for all the unknowns. Note that the equations are nonlinear. Therefore they can have many roots.

One solution can be obtained by setting s to zero to satisfy the condition $2us = 0$ in Eq. (e). Equations (b)–(d) are solved to obtain $x_1^ = x_2^* = 1$, $u^* = 1$, $s = 0$. When $s = 0$, the inequality constraint is active. x_1 , x_2 , and u are solved from the remaining three equations (b)–(d), which are linear in the variables. This is a stationary point of L , so it is a candidate minimum point. Note from Fig. 4-19 that it is actually a minimum point, since any move away from \mathbf{x}^* either violates the constraint or increases the cost function. *The second stationary point is obtained by setting $u = 0$ to satisfy the condition of Eq. (e) and solving the remaining equations for x_1 , x_2 , and s . This gives $x_1^* = x_2^* = 1.5$, $u^* = 0$, $s^2 = -1$. This is not a valid solution as the constraint is violated at the point \mathbf{x}^* , since $g = -s^2 = 1 > 0$.**

It is interesting to observe the geometrical representation of the necessary conditions for inequality constrained problems. The gradients of the cost and constraint functions at the candidate point (1, 1) are calculated as

$$\nabla f = \begin{bmatrix} 2(x_1 - 1.5) \\ 2(x_2 - 1.5) \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}; \quad \nabla g = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (\text{f})$$

These gradients are along the same line but in opposite directions as shown in Fig. 4-19. Observe also that any small move from point C either increases the cost function or takes the design into the infeasible region to reduce the cost function any further (i.e., the condition for a local minimum given in Eq. (4.1), Section 4.1.1 is violated). Thus, point (1, 1) is indeed a local minimum point. This geometrical condition is called the *sufficient condition* for a local minimum point.

It turns out that the necessary condition $u \geq 0$ ensures that the gradients of the cost and the constraint functions point in opposite directions. This way f cannot be reduced any further by stepping in the negative gradient direction without violating the constraint. That is, any further reduction in the cost function leads to leaving the feasible region at the candidate minimum point. This can be observed in Fig. 4-19.

The necessary conditions for the equality and inequality constraints can be summed up in what are commonly known as the *Karush-Kuhn-Tucker (KKT) first-order necessary conditions*, displayed in Theorem 4.6:

Theorem 4.6 Karush-Kuhn-Tucker (KKT) Optimality Conditions Let \mathbf{x}^* be a regular point of the feasible set that is a local minimum for $f(\mathbf{x})$ subject to $h_i(\mathbf{x}) = 0$; $i = 1$ to p ; $g_j(\mathbf{x}) \leq 0$; $j = 1$ to m . Then there exist Lagrange multipliers \mathbf{v}^* (a p -vector) and \mathbf{u}^* (an m -vector) such that the Lagrangian function is stationary with respect to x_j , v_i , u_j , and s_j at the point \mathbf{x}^* .

1. Lagrangian Function

$$L(\mathbf{x}, \mathbf{v}, \mathbf{u}, \mathbf{s}) = f(\mathbf{x}) + \sum_{i=1}^p v_i h_i(\mathbf{x}) + \sum_{j=1}^m u_j (g_j(\mathbf{x}) + s_j^2) = f(\mathbf{x}) + \mathbf{v}^T \mathbf{h}(\mathbf{x}) + \mathbf{u}^T (\mathbf{g}(\mathbf{x}) + \mathbf{s}^2) \quad (4.46a)$$

2. Gradient Conditions

$$\frac{\partial L}{\partial x_k} = \frac{\partial f}{\partial x_k} + \sum_{i=1}^p v_i^* \frac{\partial h_i}{\partial x_k} + \sum_{j=1}^m u_j^* \frac{\partial g_j}{\partial x_k} = 0; \quad k = 1 \text{ to } n \quad (4.46b)$$

$$\frac{\partial L}{\partial v_i} = 0 \Rightarrow h_i(\mathbf{x}^*) = 0; \quad i = 1 \text{ to } p \quad (4.47)$$

$$\frac{\partial L}{\partial u_j} = 0 \Rightarrow (g_j(\mathbf{x}^*) + s_j^2) = 0; \quad j = 1 \text{ to } m \quad (4.48)$$

3. Feasibility Check for Inequalities

$$s_j^2 \geq 0; \text{ or equivalently } g_j \leq 0; \quad j = 1 \text{ to } m \quad (4.49)$$

4. Switching Conditions

$$\frac{\partial L}{\partial s_j} = 0 \Rightarrow 2u_j^* s_j = 0; \quad j = 1 \text{ to } m \quad (4.50)$$

5. Nonnegativity of Lagrange Multipliers for Inequalities

$$u_j^* \geq 0; \quad j = 1 \text{ to } m \quad (4.51)$$

6. Regularity Check

Gradients of active constraints should be linearly independent. In such a case the Lagrange multipliers for the constraints are unique.

It is important to understand the use KKT conditions to (i) check possible optimality of a given point and (ii) determine the candidate local minimum points. Note first from Eqs. (4.47) to (4.49) that *the candidate minimum point must be feasible*, so we must check all the constraints to ensure their satisfaction. The gradient conditions of Eq. (4.46b) must also be satisfied simultaneously. These conditions have a *geometrical meaning*. To see this rewrite Eq. (4.46b) as

$$-\frac{\partial f}{\partial x_j} = \sum_{i=1}^p v_i^* \frac{\partial h_i}{\partial x_j} + \sum_{i=1}^m u_i^* \frac{\partial g_i}{\partial x_j}; \quad j = 1 \text{ to } n \quad (4.52)$$

which shows that at the stationary point, the negative gradient direction on the left side (*steepest descent direction*) for the cost function is a linear combination of the gradients of the constraints with Lagrange multipliers as the scalar parameters of the linear combination.

The m conditions in Eq. (4.50) are known as the *switching conditions* or *complementary slackness conditions*. They can be satisfied by setting either $s_i = 0$ (zero slack implies active inequality, i.e., $g_i = 0$), or $u_i = 0$ (in this case g_i must be ≤ 0 to satisfy feasibility). These conditions determine several cases in actual calculations, and their use must be clearly understood. In Example 4.29, there was only one switching condition, which gave two possible cases; case 1 where the slack variable was zero and case 2 where the Lagrange multiplier u for the inequality constraint was zero. Each of the two cases was solved for the unknowns. For general problems, there is more than one switching condition in Eq. (4.50); the number of switching conditions is equal to the number of inequality constraints for the problem. Various combinations of these conditions can give many solution cases. In general, with m inequality constraints, the switching conditions lead to 2^m distinct *normal* solution cases (abnormal case is the one where both $u_i = 0$ and $s_i = 0$). For each case, we need to solve the remaining necessary conditions for candidate local minimum points. Depending on the functions of the problem, it may or may not be possible to solve analytically the necessary conditions of each case. If the functions are nonlinear, we will have to use numerical methods to find their roots. In that case, each case may give several candidate minimum points.

We shall illustrate the use of the KKT conditions in several example problems. In Example 4.29 there were only two variables, one Lagrange multiplier and one slack variable. For general problems, the unknowns are \mathbf{x} , \mathbf{u} , \mathbf{s} , and \mathbf{v} . These are n , m , m , and p dimensional vectors. There are thus $(n + 2m + p)$ unknown variables and we need $(n + 2m + p)$ equations to determine them. The equations needed for their solution are available in the KKT necessary conditions. If we count the number of equations in Eqs. (4.46) to (4.51), we find that there are indeed $(n + 2m + p)$ equations. These equations then must be solved simultaneously for the candidate local minimum points. After the solutions are found, the remaining necessary conditions of Eqs. (4.49) and (4.51) must be checked. Conditions of Eq. (4.49) ensure feasibility of candidate local minimum points with respect to the inequality constraints $g_i(\mathbf{x}) \leq 0$; $i = 1$ to m . And, conditions of Eq. (4.51) say that the Lagrange multipliers of the “ \leq type” inequality constraints must be nonnegative.

Note that evaluation of s_i^2 essentially implies evaluation of the constraint function $g_i(\mathbf{x})$, since $s_i^2 = -g_i(\mathbf{x})$. This allows us to check feasibility of the candidate points with respect to the constraint $g_i(\mathbf{x}) \leq 0$. It is also important to note that if an inequality constraint $g_i(\mathbf{x}) \leq 0$ is inactive at the candidate minimum point \mathbf{x}^* [i.e., $g_i(\mathbf{x}^*) < 0$, or $s_i^2 > 0$], then the corresponding *Lagrange multiplier* $u_i^* = 0$ to satisfy the switching condition of Eq. (4.50). If, however, it is active [i.e., $g_i(\mathbf{x}^*) = 0$], then the Lagrange multiplier must be nonnegative, $u_i^* \geq 0$. This condition ensures that there are no feasible directions with respect to the i th constraint $g_i(\mathbf{x}^*) \leq 0$ at the candidate point \mathbf{x}^* along which the cost function can reduce any further. Stated differently, the condition ensures that any reduction in the cost function at \mathbf{x}^* can occur only by stepping into the infeasible region for the constraint $g_i(\mathbf{x}^*) \leq 0$.

Note further that the necessary conditions of Eqs. (4.46) to (4.51) are generally a nonlinear system of equations in the variables \mathbf{x} , \mathbf{u} , \mathbf{s} , and \mathbf{v} . It may not be easy to solve the system analytically. Therefore, we may have to use numerical methods such as the Newton-Raphson method of Appendix C to find roots of the system. Fortunately, software, such as Excel, MATLAB, Mathematica and others, is available in most information technology center libraries to solve a nonlinear set of equations. Such programs are of great help in solving for candidate local minimum points.

The following important points should be noted relative to the Karush-Kuhn-Tucker (KKT) first-order necessary conditions:

1. KKT conditions are *not applicable* at the points that are not *regular*. In those cases their use may yield candidate minimum points; however, the Lagrange multipliers are not unique.
2. Any point that *does not satisfy* KKT conditions *cannot be a local minimum* unless it is an irregular point (in that case KKT conditions are not applicable). Points satisfying the conditions are called KKT points.
3. *The points satisfying KKT conditions can be constrained or unconstrained*. They are unconstrained when there are no equalities and all inequalities are inactive. If the candidate point is unconstrained, it can be a local minimum, maximum, or inflection point depending on the form of the Hessian matrix of the cost function (refer to Section 4.3 for the necessary and sufficient conditions for unconstrained problems).
4. If there are equality constraints and no inequalities are active (i.e., $\mathbf{u} = \mathbf{0}$), then the points satisfying KKT conditions are only stationary. They can be minimum, maximum, or inflection points.
5. If some inequality constraints are active and their multipliers are positive, then the points satisfying KKT conditions cannot be local maxima for the cost function (they may be local maximum points if active inequalities have zero multipliers). They may not be local minima either; this will depend on the second-order necessary and sufficient conditions discussed in Chapter 5.
6. It is important to note that value of the *Lagrange multiplier* for each constraint depends on the functional form for the constraint. For example, Lagrange multiplier for the constraint $x/y - 10 \leq 0$ ($y > 0$) is different for the same constraint expressed as $x - 10y \leq 0$, or $0.1x/y - 1 \leq 0$. The optimum solution for the problem does not change by changing the form of the constraint, but its Lagrange multiplier is changed. This is further explained in Section 4.5.

Examples 4.30 and 4.31 illustrate various solutions of KKT necessary conditions for candidate local minimum points.

EXAMPLE 4.30 Various Solutions of KKT Necessary Conditions

Write KKT necessary conditions and solve them for the problem: minimize $f(\mathbf{x}) = \frac{1}{3}x^3 - \frac{1}{2}(b+c)x^2 + bcx + f_0$ subject to $a \leq x \leq d$ where $0 < a < b < c < d$ and f_0 are specified constants (created by Y. S. Ryu).

Solution. A graph for the function is shown in Fig. 4-20. It can be seen that Point A is a constrained minimum, Point B is an unconstrained maximum, Point C is an unconstrained minimum, and Point D is a constrained maximum. We shall show how the KKT conditions distinguish between these points. Note that since only one con-

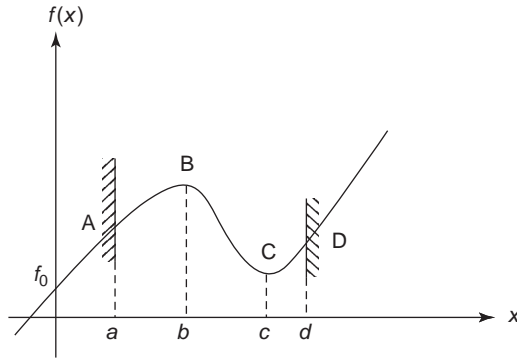


FIGURE 4-20 Graphical representation for Example 4.30. Point A, constrained local minimum; B, unconstrained local maximum; C, unconstrained local minimum; D, constrained local maximum.

straint can be active at the candidate minimum point (x cannot be at the points A and D simultaneously), all the feasible points are regular. There are two inequality constraints,

$$g_1 = a - x \leq 0; \quad g_2 = x - d \leq 0 \quad (a)$$

The Lagrangian function of Eq. (4.46a) for the problem is given as

$$L = \frac{1}{3}x^3 - \frac{1}{2}(b+c)x^2 + bcx + f_0 + u_1(a - x + s_1^2) + u_2(x - d + s_2^2) \quad (b)$$

where u_1 and u_2 are the Lagrange multipliers and s_1 and s_2 are the slack variables for $g_1 = a - x \leq 0$ and $g_2 = x - d \leq 0$, respectively. The KKT conditions give

$$\frac{\partial L}{\partial x} = x^2 - (b+c)x + bc - u_1 + u_2 = 0 \quad (c)$$

$$(a-x) + s_1^2 = 0, s_1^2 \geq 0; \quad (x-d) + s_2^2 = 0, s_2^2 \geq 0 \quad (d)$$

$$u_1 s_1 = 0; \quad u_2 s_2 = 0 \quad (e)$$

$$u_1 \geq 0; \quad u_2 \geq 0 \quad (f)$$

The switching conditions in Eq. (e) give four cases for the solution of KKT conditions. Each case will be considered separately and solved.

Case 1: $u_1 = 0, u_2 = 0$. For this case, Eq. (c) gives two solutions as $x = b$ and $x = c$. For these points both the inequalities are strictly satisfied because slack variables calculated from Eq. (d) are

$$\text{for } x = b: \quad s_1^2 = b - a > 0; \quad s_2^2 = d - b > 0 \quad (g)$$

$$\text{for } x = c: \quad s_1^2 = c - a > 0; \quad s_2^2 = d - c > 0 \quad (h)$$

Thus, all the KKT conditions are satisfied, and these are candidate minimum points. Since the points are unconstrained, they are actually stationary points. We can check the sufficient condition by calculating the *curvature* of the cost function at the two candidate points:

$$x = b; \quad \frac{d^2f}{dx^2} = 2x - (b+c) = b - c < 0 \quad (i)$$

Since $b < c$, d^2f/dx^2 is negative. Therefore, the sufficient condition for a local minimum is violated. Actually, the second-order necessary condition of Eq. (4.32) is also violated, so the point cannot be a local minimum for the function. It is actually a local maximum point because it satisfies the sufficient condition for that, as also seen in Fig. 4-20.

$$x = c; \quad \frac{d^2f}{dx^2} = c - b > 0 \quad (j)$$

Since $b < c$, d^2f/dx^2 is positive. Therefore, the second-order sufficient condition of Eq. (4.31) is satisfied, and this is a local minimum point, as also seen in Fig. 4-20.

Case 2: $u_1 = 0, s_2 = 0$. g_2 is active for this case and since $s_2 = 0$, therefore, $x = d$. Equation (c) gives

$$u_2 = -[d^2 - (b+c)d + bc] = -(d-c)(d-b) \quad (k)$$

Since $d > c > b$, u_2 is < 0 . Actually the term within the square brackets is also the slope of the function at $x = d$ which is positive, so $u_2 < 0$. The KKT necessary condition is violated, so there is no solution for this case, i.e., $x = d$ is not a candidate minimum point. This is true as can be observed for the point D in Fig. 4-20.

Case 3: $s_1 = 0, u_2 = 0$. $s_1 = 0$ implies that g_1 is active and, therefore, $x = a$. Equation (c) gives

$$u_1 = a^2 - (b+c)a + bc = (a-b)(a-c) > 0 \quad (l)$$

Also, since $u_1 =$ slope of the function at $x = a$, it is positive and all the KKT conditions are satisfied. Thus, $x = a$ is a candidate minimum point. Actually $x = a$ is a local minimum point because a feasible move from the point increases the cost function. This is a sufficient condition which we shall discuss in Chapter 5.

Case 4: $s_1 = 0, s_2 = 0$. This case for which both constraints are active does not give any valid solution since x cannot be simultaneously equal to a and d .

EXAMPLE 4.31 Solution of KKT Necessary Conditions

Solve KKT condition for the problem: minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - 3x_1x_2$ subject to $g = x_1^2 + x_2^2 - 6 \leq 0$.

Solution. The feasible region for the problem is a circle with its center at $(0, 0)$ and radius as $\sqrt{6}$. This is plotted in Fig. 4-21. Several cost function contours are shown

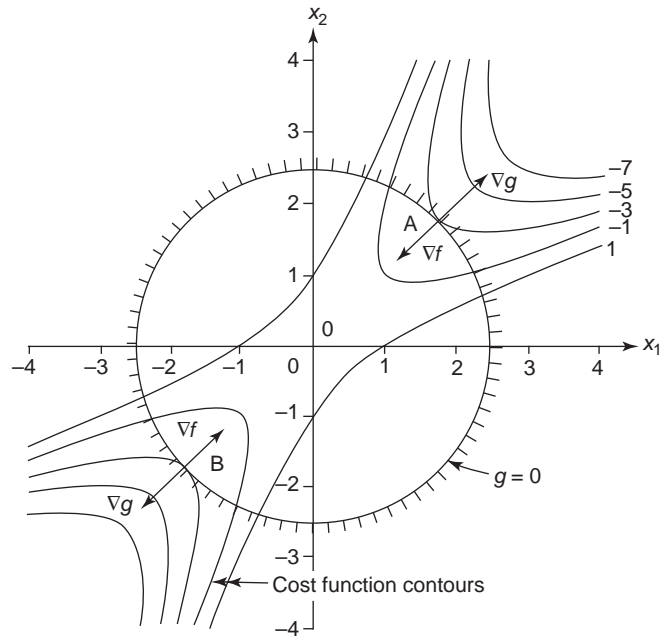


FIGURE 4-21 Graphical solution for Example 4.31. Local minimum points, A and B.

there. It can be seen that points A and B give minimum value for the cost function. The gradients of cost and constraint functions at these points are along the same line but in opposite directions, so KKT necessary conditions are satisfied. We shall verify this by writing these conditions and solving them for candidate minimum points. The Lagrange function of Eq. (4.46a) for the problem is

$$L = x_1^2 + x_2^2 - 3x_1x_2 + u(x_1^2 + x_2^2 - 6 + s^2) \quad (\text{a})$$

Since there is only one constraint for the problem, all points of the feasible region are *regular*, so the KKT necessary conditions are applicable. They are given as

$$\frac{\partial L}{\partial x_1} = 2x_1 - 3x_2 + 2ux_1 = 0 \quad (\text{b})$$

$$\frac{\partial L}{\partial x_2} = 2x_2 - 3x_1 + 2ux_2 = 0 \quad (\text{c})$$

$$x_1^2 + x_2^2 - 6 + s^2 = 0, s^2 \geq 0, u \geq 0 \quad (\text{d})$$

$$us = 0 \quad (\text{e})$$

Equations (b)–(e) are the four equations in four unknowns, x_1 , x_2 , s , and u . Thus, in principle, we have enough equations to solve for all the unknowns. The system of equations is nonlinear; however, it is possible to analytically solve for all the roots.

There are three possible ways of satisfying the switching condition of Eq. (e): (i) $u = 0$, (ii) $s = 0$, implying g is active, or (iii) $u = 0$ and $s = 0$. We will consider each case separately and solve for roots of the necessary conditions.

Case 1: $u = 0$. In this case, the inequality constraint is considered as inactive at the solution point. We shall solve for x_1 and x_2 and then check the constraint. Equations (b) and (c) reduce to

$$2x_1 - 3x_2 = 0; \quad -3x_1 + 2x_2 = 0 \quad (f)$$

This is 2×2 homogeneous system of linear equations (right side is zero). Such a system has a nontrivial solution only if the determinant of the coefficient matrix is zero. However, since the determinant of the matrix is -5 , the system has only a trivial solution, $x_1 = x_2 = 0$. We can also solve the system using Gaussian elimination procedures. This solution gives $s^2 = 6$ from Eq. (d), so the inequality is not active. Thus, the candidate minimum point for this case is

$$x_1^* = 0, x_2^* = 0, u^* = 0, f(0, 0) = 0 \quad (g)$$

Case 2: $s = 0$. In this case, $s = 0$ implies inequality as active. We must solve Eqs. (b)–(d) simultaneously for x_1 , x_2 , and u . Note that this is a nonlinear set of equations, so there can be multiple roots. Equation (b) gives $u = -1 + 3x_2/2x_1$. Substituting for u in Eq. (c), we obtain $x_1^2 = x_2^2$. Using this in Eq. (d), solving for x_1 and x_2 , and then solving for u , we obtain four roots of Eqs. (b), (c), and (d) as

$$\begin{aligned} x_1 = x_2 = \sqrt{3}, \quad u &= \frac{1}{2} \\ x_1 = x_2 = -\sqrt{3}, \quad u &= \frac{1}{2} \\ x_1 = -x_2 = \sqrt{3}, \quad u &= -\frac{5}{2} \\ x_1 = -x_2 = -\sqrt{3}, \quad u &= -\frac{5}{2} \end{aligned} \quad (h)$$

The last two roots violate KKT necessary condition, $u \geq 0$. Therefore, there are two candidate minimum points for this case. The first point corresponds to point A and the second one to B in Fig. 4-21.

Case 3: $u = 0, s = 0$. With these conditions, Eqs. (b) and (c) give $x_1 = 0, x_2 = 0$. Substituting these into Eq. (d), we obtain $s^2 = 6 \neq 0$. Therefore, all KKT conditions cannot be satisfied.

The case where both u and s are zero usually does not occur in most practical problems. This can also be explained using the *physical interpretation* of the Lagrange multipliers discussed later in this chapter. The multiplier u for a constraint $g \leq 0$ actually gives the first derivative of the cost function with respect to variation in the right side of the constraint, i.e., $u = -(\partial f/\partial e)$, where e is a small change in the constraint limit as $g \leq e$. Therefore, $u = 0$ when $g = 0$ implies that, any change in the right side of the constraint $g \leq 0$ has no effect on the optimum cost function value. This usually does not happen in practice. When the right side of a constraint is changed, the feasible region for the problem changes, which usually has some effect on the optimum solution.

Finally, the points satisfying KKT necessary conditions for the problem are summarized

1. $x_1^* = 0, x_2^* = 0, u^* = 0, f(0, 0) = 0$, Point O in Fig. 4-21
2. $x_1^* = x_2^* = \sqrt{3}, u^* = \frac{1}{2}, f(\sqrt{3}, \sqrt{3}) = -3$, Point A in Fig. 4-21
3. $x_1^* = x_2^* = -\sqrt{3}, u^* = \frac{1}{2}, f(-\sqrt{3}, -\sqrt{3}) = -3$, Point B in Fig. 4-21

It is interesting to note that points A and B satisfy the sufficient condition for local minima. As can be observed from Fig. 4-21, any feasible move from the points results in an increase in the cost and any further reduction in the cost results in violation of the constraint. It can also be observed that point O does not satisfy the sufficient condition because there are feasible directions that result in a decrease in the cost function. So, point O is only a stationary point. We shall check the sufficient conditions for this problem later in Chapter 5.

The foregoing two examples illustrate the procedure of solving Karush-Kuhn-Tucker necessary conditions for candidate local minimum points. It is extremely important to understand the procedure clearly. Example 4.31 had only one inequality constraint. The switching condition of Eq. (e) gave only two normal cases—either $u = 0$ or $s = 0$ (the abnormal case where $u = 0$ and $s = 0$ rarely gives additional candidate points, so it will be ignored). Each of the cases gave candidate minimum point \mathbf{x}^* . For case 1 ($u = 0$), there was only one point \mathbf{x}^* satisfying Eqs. (b), (c), and (d). However, for case 2 ($s = 0$), there were four roots for Eqs. (b), (c), and (d). Two of the four roots did not satisfy nonnegativity conditions on the Lagrange multipliers. Therefore, the corresponding two roots were not candidate local minimum points.

The preceding procedure is valid for more general nonlinear optimization problems. In Example 4.32, we illustrate the procedure for a problem with two design variables and two inequality constraints.

EXAMPLE 4.32 Solution of KKT Necessary Conditions

Minimize $f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 2$ subject to $g_1 = -2x_1 - x_2 + 4 \leq 0$,
 $g_2 = -x_1 - 2x_2 + 4 \leq 0$.

Solution. Figure 4-22 gives a graphical representation for the problem. The two constraint functions are plotted and the feasible region is identified. It can be seen that point $A(\frac{4}{3}, \frac{4}{3})$, where both the inequality constraints are active, is the optimum solution for the problem. Since it is a two-variable problem, only two vectors can be linearly independent. It can be seen in Fig. 4-22 that the constraint gradients ∇g_1 and ∇g_2 are linearly independent (hence the *optimum point is regular*), so any other vector can be expressed as a linear combination of them. In particular, $-\nabla f$ (the negative gradient of the cost function) can be expressed as linear combination of ∇g_1 and ∇g_2 , with positive scalars as the multipliers of the linear combination, which is precisely the KKT necessary condition of Eq. (4.46b). In the following, we shall write these conditions and solve them to verify the graphical solution.

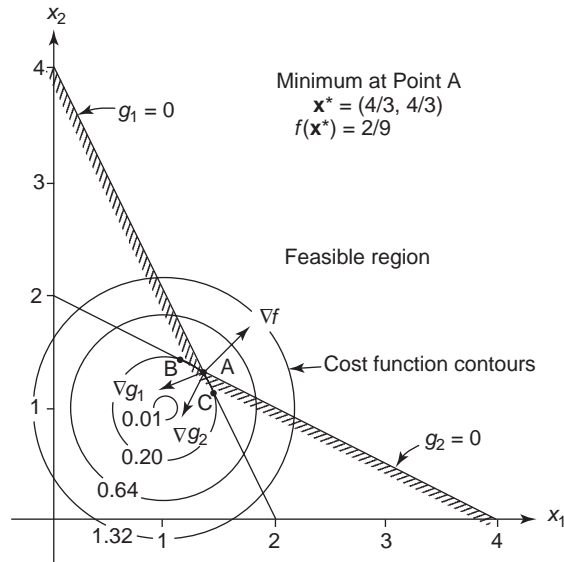


FIGURE 4-22 Graphical solution for Example 4.32.

The Lagrange function of Eq. (4.46a) for the problem is given as

$$L = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 2 + u_1(-2x_1 - x_2 + 4 + s_1^2) + u_2(-x_1 - 2x_2 + 4 + s_2^2) \quad (\text{a})$$

The KKT necessary conditions are

$$\frac{\partial L}{\partial x_1} = 2x_1 - 2 - 2u_1 - u_2 = 0 \quad (\text{b})$$

$$\frac{\partial L}{\partial x_2} = 2x_2 - 2 - u_1 - 2u_2 = 0 \quad (\text{c})$$

$$g_1 = -2x_1 - x_2 + 4 + s_1^2 = 0; \quad s_1^2 \geq 0, u_1 \geq 0 \quad (\text{d})$$

$$g_2 = -x_1 - 2x_2 + 4 + s_2^2 = 0; \quad s_2^2 \geq 0, u_2 \geq 0 \quad (\text{e})$$

$$u_i s_i = 0; \quad i = 1, 2 \quad (\text{f})$$

Equations (b)–(f) are the six equations in six unknowns: x_1 , x_2 , s_1 , s_2 , u_1 , and u_2 . We must solve them simultaneously for candidate local minimum points. One way to satisfy the switching conditions of Eq. (f) is to identify various cases and then solve them for the roots. There are four cases, and we will consider each case separately and solve for all the unknowns:

1. $u_1 = 0, \quad u_2 = 0$
2. $u_1 = 0, \quad s_2 = 0 \ (g_2 = 0)$
3. $s_1 = 0 \ (g_1 = 0), \quad u_2 = 0$
4. $s_1 = 0 \ (g_1 = 0), \quad s_2 = 0 \ (g_2 = 0)$

Case 1: $u_1 = 0, u_2 = 0$. Equations (b) and (c) give $x_1 = x_2 = 1$. This is not a valid solution as it gives $s_1^2 = -1(g_1 = 1)$, $s_2^2 = -1(g_2 = 1)$ from Eqs. (d) and (e), which implies that both inequalities are violated, and so $x_1 = 1$ and $x_2 = 1$ is not a feasible design.

Case 2: $u_1 = 0, s_2 = 0$. With these conditions, Eqs. (b), (c), and (e) become

$$2x_1 - 2 - u_2 = 0, \quad 2x_2 - 2 - 2u_2 = 0, \quad -x_1 - 2x_2 + 4 = 0 \quad (g)$$

These are three linear equations in the three unknowns x_1, x_2 , and u_2 . Any method of solving a linear system of equations such as Gaussian elimination, or method of determinants (Cramer's rule), can be used to find roots. Using the elimination procedure, we obtain $x_1 = 1.2, x_2 = 1.4$, and $u_2 = 0.4$. Therefore, the solution for this case is

$$x_1 = 1.2, x_2 = 1.4; u_1 = 0, u_2 = 0.4; f = 0.2 \quad (h)$$

We need to check for feasibility of the design point with respect to constraint g_1 before it can be claimed as a candidate local minimum point. Substituting $x_1 = 1.2$ and $x_2 = 1.4$ into Eq. (d), we find that $s_1^2 = -0.2 < 0$ ($g_1 = 0.2$), which is a violation of constraint g_1 . Therefore, case 2 also does not give any candidate local minimum point. It can be seen in Fig. 4-22 that point (1.2, 1.4) corresponds to point B, which is not in the feasible set.

Case 3: $s_1 = 0, u_2 = 0$. With these conditions Eqs. (b), (c), and (d) give

$$2x_1 - 2 - 2u_1 = 0; \quad 2x_2 - 2 - u_1 = 0; \quad -2x_1 - x_2 + 4 = 0 \quad (i)$$

This is again a linear system of equations for the variables x_1, x_2 , and u_1 . Solving the system, we get the solution as

$$x_1 = 1.4, x_2 = 1.2; u_1 = 0.4, u_2 = 0; f = 0.2 \quad (j)$$

Checking the design for feasibility with respect to constraint g_2 , we find from Eq. (e) $s_2^2 = -0.2 < 0$ ($g_2 = 0.2$). This is not a feasible design. Therefore, Case 3 also does not give any candidate local minimum point. It can be observed in Fig. 4-22 that point (1.4, 1.2) corresponds to point C, which is not in the feasible region.

Case 4: $s_1 = 0, s_2 = 0$. For this case, Eqs. (b) to (e) must be solved for the four unknowns x_1, x_2, u_1 , and u_2 . This system of equations is again linear and can be solved easily. Using the elimination procedure as before, we obtain $x_1 = \frac{4}{3}$ and $x_2 = \frac{4}{3}$ from Eqs. (d) and (e). Solving for u_1 and u_2 from Eqs. (b) and (c), we get $u_1 = \frac{2}{9} > 0$ and $u_2 = \frac{2}{9} > 0$. To check regularity condition for the point, we evaluate the gradients of the active constraints and define the constraint gradient matrix \mathbf{A} as

$$\nabla g_1 = \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \quad \nabla g_2 = \begin{bmatrix} -1 \\ -2 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} -2 & -1 \\ -1 & -2 \end{bmatrix} \quad (k)$$

Since $\text{rank}(\mathbf{A}) = \#$ of active constraints, the gradients ∇g_1 and ∇g_2 are linearly independent. Thus, all the KKT conditions are satisfied and the preceding solution is a candidate local minimum point. The solution corresponds to point A in Fig. 4-22. The cost function at the point has a value of $\frac{2}{9}$.

It can be observed in Fig. 4-22 that the vector $-\nabla f$ can be expressed as a linear combination of the vectors ∇g_1 and ∇g_2 at point A. This satisfies the necessary condition of Eq. (4.52). It can also be seen from the figure that point A is indeed a local minimum because any further reduction in the cost function is possible only if we go into the infeasible region. Any feasible move from point A results in an increase in the cost function.

Note that addition of an inequality to the problem formulation doubles the number of KKT solution cases. With 2 inequalities, we had 4 KKT cases; with 3 inequalities we will have 8 cases; and with 4 inequalities, we will have 16 cases. Therefore the number of cases quickly gets out of hand and thus this solution procedure cannot be used to solve most practical problems. Based on these conditions, however, numerical methods have been developed that can handle any number of equality and inequality constraints. In Section 4.7, we shall solve two problems having 16 and 32 cases, respectively. *In summary, the following points should be noted regarding Karush-Kuhn-Tucker first-order necessary conditions:*

1. *The conditions can be used to check* whether a given point is a candidate minimum; it must be feasible, the gradient of the Lagrangian with respect to the design variables must be zero, and the Lagrange multipliers for inequality constraints must be nonnegative.
2. For a given problem, *the conditions can be used to find* candidate minimum points. Several cases defined by the switching conditions must be considered and solved. Each case can give multiple solutions.
3. For each solution case, *remember to*
 - (i) check all inequality constraints for feasibility (i.e., $g_i \leq 0$ or $s_i^2 \geq 0$)
 - (ii) calculate all the Lagrange multipliers
 - (iii) ensure that the Lagrange multipliers for all the inequality constraints are nonnegative

4.4.4 Solution of KKT Conditions Using Excel

Excel Solver was introduced in Section 4.3.4 to find roots of a nonlinear equation. We shall use that capability to solve the KKT conditions for the problem solved in Example 4.31. The first step in the solution process is to prepare the Excel worksheet to describe the problem functions. Then Solver is invoked under the Tools menu to define equations and constraints. Figure 4-23 shows the worksheet for the problem and the Solver Parameters dialog box. Cells A5 to A8 show the variable names that will appear later in the “Answer Report” worksheet. Cells B5 to B8 are named as x, y, u and s, respectively and contain the starting values for the four variables. Note that the variables x_1 and x_2 have been changed to x and y because x_1 and x_2 are not valid names in Excel. Cells A10 to A15 contain expressions for the KKT conditions given in Eqs. (b) to (e) in Example 4.31. These expressions will appear later in the “Answer Report.” Cells B10 to B15 contain the expressions coded in terms of the variable cells B4 to B7 as follows:

Cell B10: = $2*x-3*y+2*u*s$ (expression for $\partial L/\partial x$)
 Cell B11: = $2*y-3*x+2*u*y$ (expression for $\partial L/\partial y$)
 Cell B12: = $x*x+y*y-6+s*s$ (constraint, $g + s^2$)
 Cell B13: = $u*s$ (switching condition)
 Cell B14: = $s*s$ (s^2)
 Cell B15: = u (u)

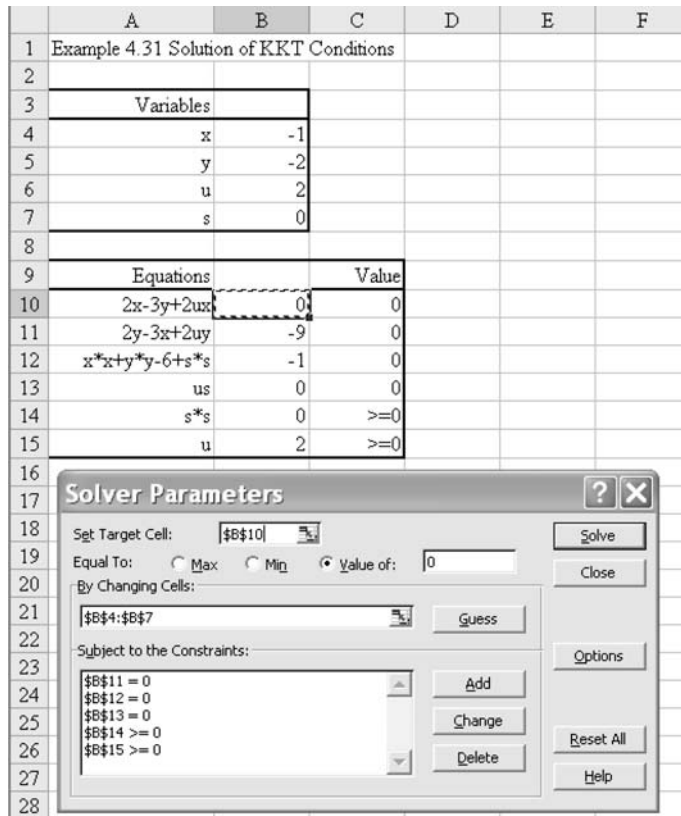


FIGURE 4-23 Excel Worksheet and Solver Parameters dialog box for Example 4.31.

The current values for these cells for starting values of the variables are shown in Fig. 4-23. Now the root finding problem can be defined in the “Solver Parameters” dialog box. The target cell is set to B10, whose value is set to zero at the solution point. The variable cells are identified as B5 to B8. The rest of the equations are entered as constraints by clicking the “Add” button. Note that in order to solve a set of nonlinear equations, one of the equations is identified as the target equation (#1 in the present case), and rest of them are identified as constraints. Once the problem has been defined, the “Solve” button is clicked to solve the problem. Solver solves the problem and reports the final results by updating the original worksheet and opening the “Solver Results” dialog box, as shown in Fig. 4-24. The final “Answer” worksheet can be generated if desired. The current starting point of (1, -2, 2, 0) gave the KKT point as (-1.732, -1.732, 0.5, 0).

It is important to note that using the worksheet shown in Fig. 4-23, the two KKT cases can be solved. These cases can be generated using starting values for the slack variable and the Lagrange multiplier. For example, selecting $u = 0$ and $s > 0$ generates the case where the constraint is inactive. This gives the solution $x = 0$ and $y = 0$. Selecting $u > 0$ and $s = 0$ gives the case where the constraint is active. Selecting different starting values for x and y gives two other points as solutions of the necessary conditions. When there are two or more inequality constraints, various KKT cases can be generated in a similar way.

4.4.5 Solution of KKT Conditions Using MATLAB

MATLAB can also be used to solve a set of nonlinear equations. The primary command used for this purpose is *fsolve*. This command is part of MATLAB Optimization Toolbox which

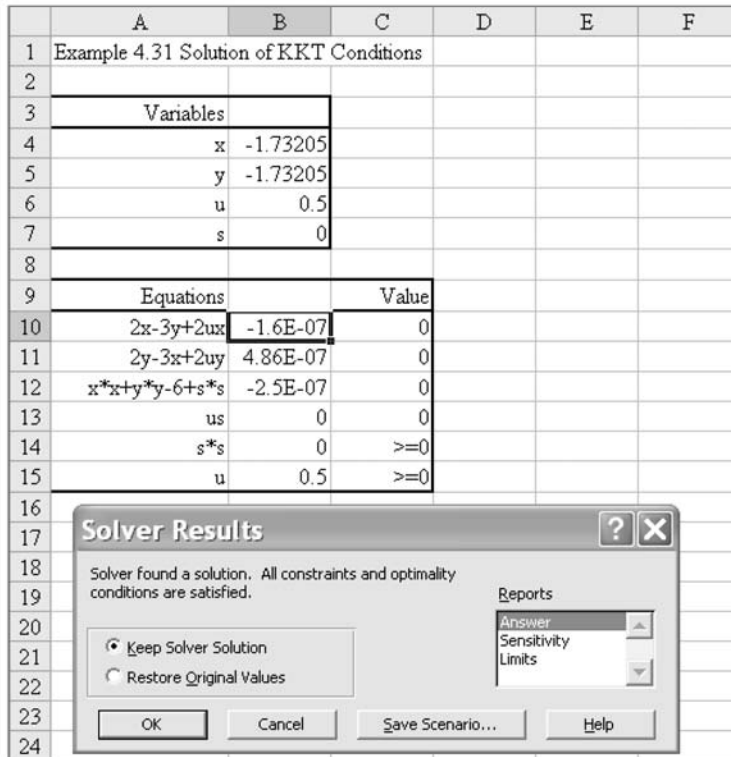


FIGURE 4-24 Solver Results for Example 4.31.

must also be installed in the computer. We shall discuss use of this capability by solving the KKT conditions for the problem of Example 4.31. When using MATLAB, it is necessary first to create a separate M-file containing the equations in the form $\mathbf{F}(\mathbf{x}) = \mathbf{0}$. For the present example, components of the vector \mathbf{x} are defined as $x(1) = x_1$, $x(2) = x_2$, $x(3) = u$, and $x(4) = s$. In terms of these variables, the KKT conditions of Eqs. (b) to (e) in Example 4.31 are given as

$$2*x(1) - 3*x(2) + 2*x(3)*x(1) = 0$$

$$2*x(2) - 3*x(1) + 2*x(3)*x(2) = 0$$

$$x(1)^2 + x(2)^2 - 6 + x(4)^2 = 0$$

$$x(3)*x(4) = 0$$

The file defining the equations is prepared as follows:

```
Function F = kktssystem(x)
F = [2*x(1) - 3*x(2) + 2*x(3)*x(1);
2*x(2) - 3*x(1) + 2*x(3)*x(2);
x(1)^2 + x(2)^2 - 6 + x(4)^2;
x(3)*x(4)];
```

The first line defines a function, named “kktsystem,” that accepts a vector of variables \mathbf{x} and returns a vector of function values \mathbf{F} . This file should be named “kktsystem” (the same name as the function itself), and as with other MATLAB files, it should be saved with a suffix of “.m.” Next, the main commands are entered interactively or in a separate file as follows:

```
x0=[1;1;1];
options=optimset('Display','iter')
x=fsolve(@kktsystem,x0,options)
```

\mathbf{x}_0 is the starting point or initial guess. The “options” command displays output for each iteration. If the command `Options = optimset('Display','off')` is used, then only the final solution is provided. The command “fsolve” finds a root of the system of equations provided in the function “kktsystem.” Although there may be many potential solutions, the solution closest to the initial guess is provided. Consequently, different starting points must be used to find different points that satisfy the KKT conditions. Starting with the given point, the solution is obtained as (1.732, 1.732, 0.5, 0).

4.5 Postoptimality Analysis: Physical Meaning of Lagrange Multipliers

The study of variations in the optimum solution as some of the original problem parameters are changed is known as *postoptimality analysis* or *sensitivity analysis*. This is an important topic for optimum design of engineering systems. Variation of the optimum cost function and design variables due to the variations of many parameters can be studied. Since sensitivity of the cost function to the variations in the constraint limit values can be studied without any further analysis, we shall focus on this aspect of sensitivity analysis only. We shall assume that the minimization problem has been solved with $h_i(\mathbf{x}) = 0$ and $g_j(\mathbf{x}) \leq 0$, i.e., with the current limit values for the constraints as zero. Thus, we like to know what happens to the optimum cost function when the constraint limits are changed from zero.

It turns out that the Lagrange multipliers (\mathbf{v}^* , \mathbf{u}^*) at the optimum design provide information to answer the foregoing sensitivity question. The investigation of this question leads to a physical interpretation of the Lagrange multipliers that can be very useful in practical applications. The interpretation will also show why the Lagrange multipliers for the “ \leq type” constraints have to be nonnegative. The multipliers show the *benefit of relaxing a constraint* or *the penalty associated with tightening it*; relaxation enlarges the feasible set, while tightening contracts it. The sensitivity result is stated in a theorem. Later in this section we shall also discuss what happens to the Lagrange multipliers if the cost and constraint functions for the problem are scaled.

4.5.1 Effect of Changing Constraint Limits

To discuss changes in the cost function due to changes in the constraint limits, we consider the modified problem of minimizing $f(\mathbf{x})$ subject to the constraints

$$h_i(\mathbf{x}) = b_i; i = 1 \text{ to } p \text{ and } g_j(\mathbf{x}) \leq e_j; j = 1 \text{ to } m \quad (4.53)$$

where b_i and e_j are small variations in the neighborhood of zero. It is clear that the optimum point for the perturbed problem depends on vectors \mathbf{b} and \mathbf{e} , i.e., it is a function of \mathbf{b} and \mathbf{e} that can be written as $\mathbf{x}^* = \mathbf{x}^*(\mathbf{b}, \mathbf{e})$. Also, optimum cost function value depends on \mathbf{b} and \mathbf{e} , i.e., $f = f(\mathbf{b}, \mathbf{e})$. However, explicit dependence of the cost function on \mathbf{b} and \mathbf{e} is not known,

i.e., an expression for f in terms of b_i and e_j cannot be obtained. The following theorem gives a way of obtaining the partial derivatives $\partial f/\partial b_i$ and $\partial f/\partial e_j$.

Theorem 4.7 Constraint Variation Sensitivity Theorem Let $f(\mathbf{x})$, $h_i(\mathbf{x})$, $i = 1$ to p , and $g_j(\mathbf{x})$, $j = 1$ to m , have two continuous derivatives. Let \mathbf{x}^* be a regular point that, together with the multipliers v_i^* and u_j^* satisfies both the KKT necessary conditions and the sufficient conditions presented in the next chapter for an isolated local minimum point for the problem defined in Eqs. (4.37) to (4.39). If for each $g_j(\mathbf{x}^*)$, it is true that $u_j^* > 0$, then the solution $\mathbf{x}^*(\mathbf{b}, \mathbf{e})$ of the modified optimization problem defined in Eq. (4.53) is a continuously differentiable function of \mathbf{b} and \mathbf{e} in some neighborhood of $\mathbf{b} = \mathbf{0}$, $\mathbf{e} = \mathbf{0}$. Furthermore,

$$\frac{\partial f(\mathbf{x}^*(\mathbf{0}, \mathbf{0}))}{\partial b_i} = -v_i^*; \quad i = 1 \text{ to } p \quad \text{and} \quad \frac{\partial f(\mathbf{x}^*(\mathbf{0}, \mathbf{0}))}{\partial e_j} = -u_j^*; \quad j = 1 \text{ to } m \quad (4.54)$$

The theorem gives values for implicit first-order derivatives of the cost function f with respect to the right side parameters of the constraints b_i and e_j . The derivatives can be used to calculate changes in the cost function as b_i and e_j are changed. Note that the theorem is applicable only when the inequality constraints are written in the “ \leq ” form. Using the theorem we can estimate changes in the cost function if we decide to adjust the right side of constraints in the neighborhood of zero. For this purpose, Taylor’s expansion for the cost function in terms of b_i and e_j can be used. Let us assume that we want to vary the right sides, b_i and e_j , of i th equality and j th inequality constraints. First-order Taylor’s expansion for the cost function about the point $b_i = 0$ and $e_j = 0$ and is given as

$$f(b_i, e_j) = f(0, 0) + \frac{\partial f(0, 0)}{\partial b_i} b_i + \frac{\partial f(0, 0)}{\partial e_j} e_j$$

Or, substituting from Eq. (4.54), we obtain

$$f(b_i, e_j) = f(0, 0) - v_i^* b_i - u_j^* e_j \quad (4.55)$$

where $f(0, 0)$ is the optimum cost function value obtained with $b_i = 0$, and $e_j = 0$. Using Eq. (4.55), a first-order change in the cost function δf due to small changes in b_i and e_j is given as

$$\delta f^* = f(b_i, e_j) - f(0, 0) = -v_i^* b_i - u_j^* e_j \quad (4.56)$$

For given values of b_i and e_j we can estimate the new value of the cost function from Eq. (4.55). If we want to change the right side of more constraints, we simply include them in Eq. (4.56) and obtain the change in cost function as

$$\delta f^* = -\sum v_i^* b_i - \sum u_j^* e_j \quad (4.57)$$

It is useful to note that if conditions of Theorem 4.7 are not satisfied, existence of implicit derivatives of Eq. (4.54) is not ruled out by the theorem. That is, the derivatives may still exist but their existence cannot be guaranteed by Theorem 4.7. This observation shall be verified later in an example problem in Section 4.7.2.

Equation (4.56) can also be used to show that the *Lagrange multiplier corresponding to a “ \leq type” constraint must be nonnegative*. To see this, let us assume that we want to relax an inequality constraint $g_j \leq 0$ that is active ($g_j = 0$) at the optimum point, i.e., we select $e_j >$

0 in Eq. (4.53). When a constraint is relaxed, the feasible set for the design problem expands. We allow more feasible designs to be candidate minimum points. Therefore, with the expanded feasible set we expect the optimum cost function to reduce further or at the most remain unchanged (Example 4.33). We observe from Eq. (4.56) that if $u_j^* < 0$, then relaxation of the constraint ($e_j > 0$) results in an increase in cost ($\delta f = -u_j^* e_j > 0$). This is a contradiction as it implies that there is a penalty to relax the constraint. Therefore, the *Lagrange multiplier for a “ \leq type” constraint must be nonnegative*.

EXAMPLE 4.33 Effect of Variations of Constraint Limits on Optimum Cost Function

To illustrate the use of constraint variation sensitivity theorem, we consider the following problem solved as Example 4.31 and discuss the effect of changing the limit for the constraint: minimize

$$f(x_1, x_2) = x_1^2 + x_2^2 - 3x_1x_2 \text{ subject to } g(x_1, x_2) = x_1^2 + x_2^2 - 6 \leq 0. \quad (\text{a})$$

Solution. The graphical solution for the problem is given in Fig. 4-21. A point satisfying both necessary and sufficient conditions is

$$x_1^* = x_2^* = \sqrt{3}, \quad u^* = \frac{1}{2}, \quad f(\mathbf{x}^*) = -3 \quad (\text{b})$$

We like to see what happens if we change the right side of the constraint equation to a value “ e ” from zero. Note that the constraint $g(x_1, x_2) \leq 0$ gives a circular feasible region with its center at (0,0) and its radius as $\sqrt{6}$, as shown in Fig. 4-21. From Theorem 4.7, we have

$$\frac{\partial f(\mathbf{x}^*)}{\partial e} = -u^* = -\frac{1}{2} \quad (\text{c})$$

If we set $e = 1$, the new value of cost function will be approximately $-3 + (-\frac{1}{2})(1) = -3.5$ using Eq. (4.55). This is consistent with the new feasible set because with $e = 1$, the radius of the circle becomes $\sqrt{7}$ and the feasible region is expanded (as can be seen in Fig. 4-21). We should expect some reduction in the cost function. If we set $e = -1$, then the effect is opposite. The feasible set becomes smaller and the cost function increases to -2.5 using Eq. (4.55).

From the foregoing discussion and example, we see that *optimum Lagrange multipliers give very useful information about the problem*. The designer can compare the magnitude of the multipliers for the active constraints. The multipliers with relatively larger values will have a significant effect on optimum cost if the corresponding constraints are changed. *The larger the value of the Lagrange multiplier, the higher is the dividend to relax the constraint, or the higher is the penalty to tighten the constraint*. Knowing this, the designer can select a few critical constraints having the greatest influence on the cost function, and then analyze to see if these constraints can be relaxed to further reduce the optimum cost function value.

4.5.2 Effect of Cost Function Scaling on Lagrange Multipliers

On many occasions, a cost function for the problem is multiplied by a positive constant. As noted in Section 4.3, any scaling of the cost function does not alter the optimum point. It does, however, change the optimum value for the cost function. The scaling should also affect the implicit derivatives of Eqs. (4.54) for the cost function with respect to the right side parameters of the constraints. *We observe from these equations that all the Lagrange multipliers also get multiplied by the same constant.* Let u_j^* and v_i^* be the Lagrange multipliers for inequality and equality constraints, respectively, and $f(\mathbf{x}^*)$ be the optimum value of the cost function at the solution point \mathbf{x}^* . Let the cost function be scaled as $\bar{f}(\mathbf{x}) = Kf(\mathbf{x})$, where $K > 0$ is a given constant, and \bar{u}_j^* and \bar{v}_i^* be the optimum Lagrange multipliers for the inequality and equality constraints, respectively, for the changed problem. Then the optimum design variable vector for the perturbed problem is \mathbf{x}^* and the relationship between optimum Lagrange multipliers is derived using the KKT conditions for the original and the changed problems, as

$$\bar{u}_j^* = Ku_j^* \quad \text{and} \quad \bar{v}_i^* = Kv_i^* \quad (4.58)$$

Example 4.34 shows the effect of scaling the cost function on the Lagrange multipliers.

EXAMPLE 4.34 Effect of Scaling the Cost Function on the Lagrange Multipliers

Consider Example 4.31: minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - 3x_1x_2$ subject to $g(\mathbf{x}) = x_1^2 + x_2^2 - 6 \leq 0$. Study the effect on the optimum solution of scaling the cost function by a constant $K > 0$.

Solution. The graphical solution for the problem is given in Fig. 4-21. A point satisfying both the necessary and sufficient condition is

$$x_1^* = x_2^* = \sqrt{3}, \quad u^* = \frac{1}{2}, \quad f(\mathbf{x}^*) = -3 \quad (a)$$

Let us solve the scaled problem by writing KKT conditions. The Lagrangian for the problem is given as (quantities with an over bar are for the perturbed problem):

$$L = K(x_1^2 + x_2^2 - 3x_1x_2) + \bar{u}(x_1^2 + x_2^2 - 6 + \bar{s}^2) \quad (b)$$

The necessary conditions give

$$\frac{\partial L}{\partial x_1} = 2Kx_1 - 3Kx_2 + 2\bar{u}x_1 = 0 \quad (c)$$

$$\frac{\partial L}{\partial x_2} = 2Kx_2 - 3Kx_1 + 2\bar{u}x_2 = 0 \quad (d)$$

$$x_1^2 + x_2^2 - 6 + \bar{s}^2 = 0; \quad \bar{s}^2 \geq 0 \quad (e)$$

$$\bar{u}\bar{s} = 0, \quad \bar{u} \geq 0 \quad (f)$$

As in Example 4.31, the case where $\bar{s} = 0$ gives candidate minimum points. Solving Eqs. (c)–(e), we get the two KKT points as

$$x_1^* = x_2^* = \sqrt{3}, \quad \bar{u}^* = K/2, \quad \bar{f}(\mathbf{x}^*) = -3K \quad (\text{g})$$

$$x_1^* = x_2^* = -\sqrt{3}, \quad \bar{u}^* = K/2, \quad \bar{f}(\mathbf{x}^*) = -3K \quad (\text{h})$$

Therefore, comparing the solutions with those obtained in Example 4.31, we observe that $\bar{u}^* = Ku^*$.

4.5.3 Effect of Scaling a Constraint on Its Lagrange Multiplier

Many times, a constraint is scaled by a positive constant. We would like to know the effect of this scaling on the Lagrange multiplier for the constraint. *It should be noted that scaling of a constraint does not change the constraint boundary, so it has no effect on the optimum solution. Only the Lagrange multiplier for the scaled constraint is affected. Looking at the implicit derivatives of the cost function with respect to the constraint right side parameters, we observe that the Lagrange multiplier for the scaled constraint gets divided by the scaling parameter.* Let $M_j > 0$ and P_i be the scale parameters for the j th inequality and i th equality constraints ($\bar{g}_j = M_j g_j$; $\bar{h}_i = P_i h_i$), and u_j^* and v_i^* , and \bar{u}_j^* and \bar{v}_i^* the corresponding Lagrange multipliers for the original and the scaled constraints, respectively. Then the following relations hold for the Lagrange multipliers:

$$\bar{u}_j^* = u_j^*/M_j \quad \text{and} \quad \bar{v}_i^* = v_i^*/P_i \quad (4.59)$$

Example 4.35 illustrates the effect of scaling a constraint on its Lagrange multiplier.

EXAMPLE 4.35 Effect of Scaling a Constraint on its Lagrange Multiplier

Consider Example 4.31 and study the effect of multiplying the inequality by a constant $M > 0$.

Solution. The Lagrange function for the problem with scaled constraint is given as

$$L = x_1^2 + x_2^2 - 3x_1x_2 + \bar{u}[M(x_1^2 + x_2^2 - 6) + \bar{s}^2] \quad (\text{a})$$

The KKT conditions give

$$\frac{\partial L}{\partial x_1} = 2x_1 - 3x_2 + 2\bar{u}Mx_1 = 0 \quad (\text{b})$$

$$\frac{\partial L}{\partial x_2} = 2x_2 - 3x_1 + 2\bar{u}Mx_2 = 0 \quad (\text{c})$$

$$M(x_1^2 + x_2^2 - 6) + \bar{s}^2 = 0; \quad \bar{s}^2 \geq 0 \quad (\text{d})$$

$$\bar{u}\bar{s} = 0, \quad \bar{u} \geq 0 \quad (\text{e})$$

As in Example 4.31, only the case with $\bar{s} = 0$ gives candidate optimum points. Solving this case, we get the two KKT points:

$$x_1^* = x_2^* = \sqrt{3}, \quad \bar{u}^* = \frac{1}{2M}, \quad f(\mathbf{x}^*) = -3 \quad (\text{f})$$

$$x_1^* = x_2^* = \sqrt{3}, \quad \bar{u}^* = \frac{1}{2M}, \quad f(\mathbf{x}^*) = -3 \quad (\text{g})$$

Therefore, comparing these solutions with the ones for Example 4.31, we observe that $\bar{u}^* = \bar{u}^*/M$.

4.5.4 Generalization of Constraint Variation Sensitivity Result

Many times variations are desired with respect to parameters that are embedded in the constraint expression in a complex way. Therefore the sensitivity expressions given in Eq. (4.54) need to be generalized. We shall pursue these generalizations for the inequality constraints only in the following paragraphs; equality constraints can be treated in similar ways. It turns out that the sensitivity of the optimum cost function with respect to an inequality constraint can be written as

$$\frac{\partial f(\mathbf{x}^*)}{\partial g_j} = u_j^*, \quad j = 1 \text{ to } m \quad (4.60)$$

If the constraint function depends on a parameter s as $g_j(s)$, then variation with respect to the parameter s can be written using the chain rule of differentiation as

$$\frac{df(\mathbf{x}^*)}{ds} = \frac{\partial f(\mathbf{x}^*)}{\partial g_j} \frac{dg_j}{ds} = u_j^* \frac{dg_j}{ds} \quad (4.61)$$

Therefore change in the cost function due to a small change δs in the parameter s is given as

$$\delta f^* = \frac{df}{ds} \delta s = u_j^* \frac{dg_j}{ds} \delta s \quad (4.62a)$$

Another way of writing this small change to the cost function would be to express it in terms of changes to constraint function itself, using Eq. (4.60) as

$$\delta f^* = \frac{df}{dg_j} \delta g_j = u_j^* \delta g_j \quad (4.62b)$$

Sometimes the right side e_j is dependent on a parameter s . In that case sensitivity of the cost function f with respect to s (derivative of f with respect to s) can be obtained directly from Eq. (4.54) using the chain rule of differentiation as

$$\frac{df(\mathbf{x}^*)}{ds} = \frac{\partial f(\mathbf{x}^*)}{\partial e_j} \frac{de_j}{ds} = -u_j^* \frac{de_j}{ds} \quad (4.63)$$

4.6 Global Optimality

In the optimum design of systems, the question about global optimality of a solution always arises. In general, it is difficult to answer the question satisfactorily. However, an answer can be attempted in the following two ways:

1. If the cost function $f(\mathbf{x})$ is continuous on a closed and bounded feasible set, then Weierstrauss Theorem 4.1 guarantees the existence of a global minimum. Therefore, if we calculate all the local minimum points, then the point that gives the least value to the cost function can be selected as a global minimum for the function. This is called exhaustive search.
2. If the optimization problem can be shown to be convex, then any local minimum is also a global minimum; also the KKT necessary conditions are sufficient for the minimum point.

Both these procedures can involve substantial computations. In this section we pursue the second approach and discuss topics of *convexity* and *convex programming problems*. Such problems are defined in terms of convex sets and convex functions; specifically convexity of the feasible set and the cost function. Therefore, we introduce these concepts and discuss results regarding global optimum solutions.

4.6.1 Convex Sets

A convex set S is a collection of points (vectors \mathbf{x}) having the following property: If P_1 and P_2 are any points in S , then the entire line segment P_1-P_2 is also in S . This is a necessary and sufficient condition for convexity of the set S . Figure 4-25 shows some examples of convex and nonconvex sets. To explain convex sets further, let us consider points on a real line along the x -axis (Fig. 4-26). Points in any interval on the line represent a convex set. Consider an interval between points a and b as shown in Fig. 4-26. To show that it is a convex set, let x_1 and x_2 be two points in the interval. The line segment between the points can be written as

$$x = \alpha x_2 + (1 - \alpha)x_1; \quad 0 \leq \alpha \leq 1 \quad (4.64)$$

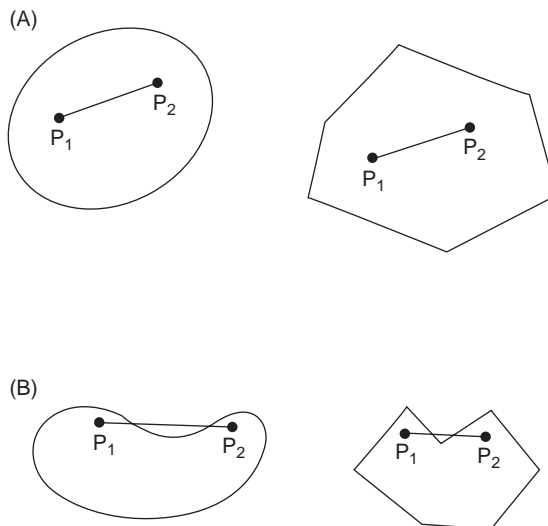


FIGURE 4-25 (A) Convex sets. (B) Nonconvex sets.

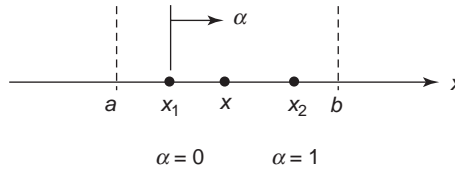


FIGURE 4-26 Convex interval between a and b on a real line.

In this equation, if $\alpha = 0$, $x = x_1$ and if $\alpha = 1$, $x = x_2$. It is clear that the line defined in Eq. (4.64) is in the interval $[a, b]$. In general, for the n -dimensional space, the *line segment* between any two points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ can be written as

$$\mathbf{x} = \alpha \mathbf{x}^{(2)} + (1 - \alpha) \mathbf{x}^{(1)}; \quad 0 \leq \alpha \leq 1 \quad (4.65)$$

If the entire line segment of Eq. (4.65) is in the set S , then it is a convex set. Equation (4.65) is a generalization of Eq. (4.64) and is called the *parametric representation of a line segment* between the points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$. A check of the convexity of a set is demonstrated in Example 4.36.

EXAMPLE 4.36 Check for Convexity of a Set

Show convexity of the set

$$S = \{\mathbf{x} \mid x_1^2 + x_2^2 - 1.0 \leq 0\}$$

Solution. To show the set S graphically, we first plot the constraint as an equality that represents a circle of radius 1 centered at $(0, 0)$, shown in Fig. 4-27. Points inside or on the circle are in S . Geometrically we see that for any two points inside the circle, the line segment between them is also inside the circle. Therefore, S is a convex set. We can also use Eq. (4.65) to show convexity of S . To do this take any two points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ in the set S . Use of Eq. (4.65) to calculate \mathbf{x} and the condition that the distance between $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ is nonnegative (i.e., $\|\mathbf{x}^{(1)} - \mathbf{x}^{(2)}\| \geq 0$), will show $\mathbf{x} \in S$. This will prove the convexity of S and is left as an exercise. Note that if the foregoing set S is defined by reversing the inequality as $x_1^2 + x_2^2 - 1.0 \geq 0$, then it will consist of points outside the circle. Such a set is clearly nonconvex because it violates the condition that the line segment of Eq. (4.65) defined by any two points in the set is not entirely in the set.

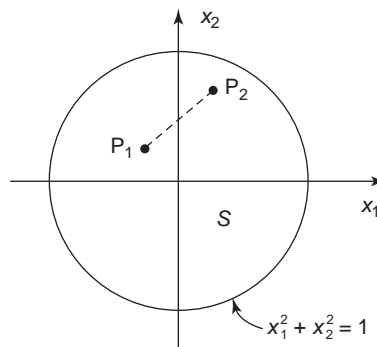


FIGURE 4-27 Convex set S for Example 4.36.

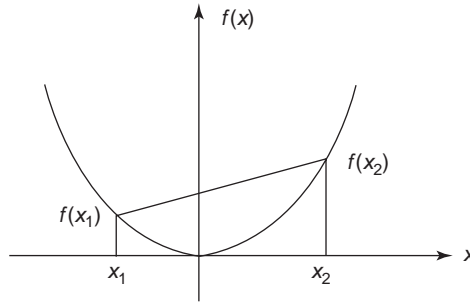


FIGURE 4-28 Convex function $f(x) = x^2$.

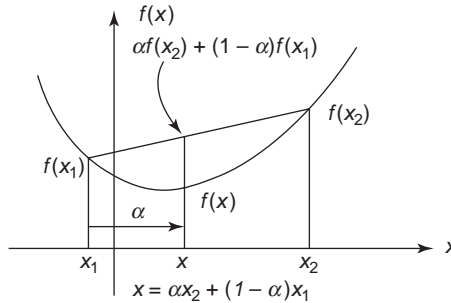


FIGURE 4-29 Characterization of a convex function.

4.6.2 Convex Functions

Consider a function of single variable $f(x) = x^2$. Graph of the function is shown in Fig. 4-28. Note that if a straight line is constructed between any two points $(x_1, f(x_1))$ and $(x_2, f(x_2))$ on the curve, the line lies above the graph of $f(x)$ at all points between x_1 and x_2 . This property characterizes convex functions.

The convex function of a single variable $f(x)$ is defined on a convex set, i.e., the independent variable x must lie in a convex set. A function $f(x)$ is called convex on the convex set S if the graph of the function lies below the line joining any two points on the curve $f(x)$. Figure 4-29 shows geometrical representation of a convex function. Using the geometry, the foregoing definition of a convex function can be expressed by the inequality $f(x) \leq \alpha f(x_2) + (1 - \alpha)f(x_1)$. Since $x = \alpha x_2 + (1 - \alpha)x_1$, the inequality becomes

$$f(\alpha x_2 + (1 - \alpha)x_1) \leq \alpha f(x_2) + (1 - \alpha)f(x_1) \quad \text{for } 0 \leq \alpha \leq 1 \quad (4.66)$$

The definition can be generalized to functions of n variables. A function $f(\mathbf{x})$ defined on a convex set S is convex if it satisfies the inequality

$$f(\alpha \mathbf{x}^{(2)} + (1 - \alpha)\mathbf{x}^{(1)}) \leq \alpha f(\mathbf{x}^{(2)}) + (1 - \alpha)f(\mathbf{x}^{(1)}) \quad \text{for } 0 \leq \alpha \leq 1 \quad (4.67)$$

for any two points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ in S . Note that convex set S is a region in the n -dimensional space satisfying the convexity condition. Equations (4.66) and (4.67) give *necessary and sufficient conditions for convexity of a function*. However, they are difficult to use in practice because we will have to check an infinite number of pairs of points. Fortunately, the following theorem gives an easier way of checking the convexity of a function.

Theorem 4.8 Check for Convexity of a Function A function of n variables $f(x_1, x_2, \dots, x_n)$ defined on a convex set S is convex if and only if the Hessian matrix of the function is positive semidefinite or positive definite at all points in the set S . If the Hessian matrix is positive definite for all points in the feasible set, then f is called a *strictly convex function*. (Note that the converse of this is not true, i.e., a strictly convex function may have only positive semidefinite Hessian at some points; e.g., $f(x) = x^4$ is a strictly convex function but its second derivative is zero at $x = 0$.)

Note that the Hessian condition of Theorem 4.8 is both necessary and sufficient, i.e., the function is not convex if the Hessian is not at least positive semidefinite for all points in the set S . Therefore if it can be shown that the Hessian is not positive definite or positive semidefinite at some points in the set S , then the function is not convex because the condition of the Theorem 4.8 is violated. In one dimension, the convexity check of the theorem reduces to the condition that the second derivative (curvature) of the function be nonnegative. The graph of such a function has nonnegative curvature, as for the functions in Figs. 4-28 and 4-29. The theorem can be proved by writing a Taylor's expansion for the function $f(\mathbf{x})$ and then using the definition of Eqs. (4.66) and (4.67). Examples 4.37 and 4.38 illustrate the check for convexity of functions.

EXAMPLE 4.37 Check for Convexity of a Function

$$f(\mathbf{x}) = x_1^2 + x_2^2 - 1$$

Solution. The domain for the function (which is all values of x_1 and x_2) is convex. The gradient and Hessian of the function are given as

$$\nabla f = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

By either of the tests given in Theorems 4.2 and 4.3 ($M_1 = 2, M_2 = 4, \lambda_1 = 2, \lambda_2 = 2$), we see that \mathbf{H} is positive definite everywhere. Therefore, f is a strictly convex function.

EXAMPLE 4.38 Check for Convexity of a Function

$$f(x) = 10 - 4x + 2x^2 - x^3$$

Solution. The second derivative of the function is $d^2f/dx^2 = 4 - 6x$. For the function to be convex, $d^2f/dx^2 \geq 0$. Thus, the function is convex only if $4 - 6x \geq 0$ or $x \leq \frac{2}{3}$. The convexity check actually defines a domain for the function over which it is convex. The function $f(x)$ is plotted in Fig. 4-30. It can be seen that the function is convex for $x \leq \frac{2}{3}$ and concave for $x \geq \frac{2}{3}$ [a function $f(x)$ is called *concave* if $-f(x)$ is convex].

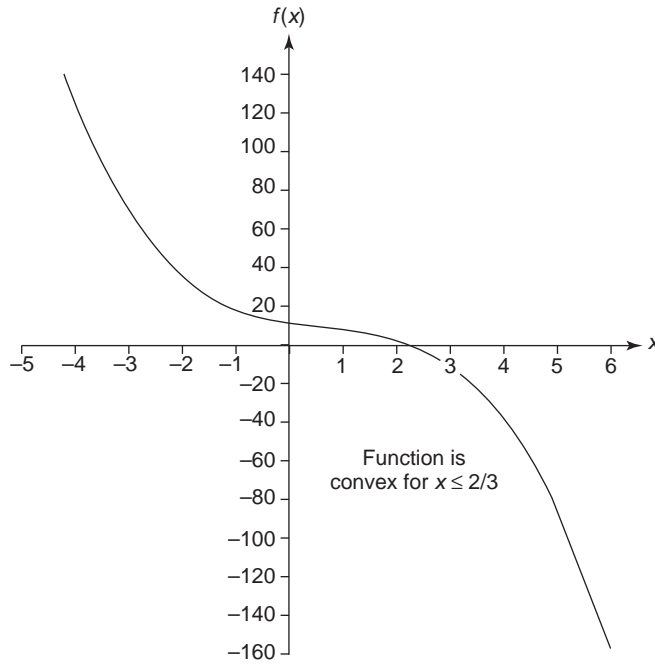


FIGURE 4-30 Graph of the function $f(x) = 10 - 4x + 2x^2 - x^3$ of Example 4.38.

4.6.3 Convex Programming Problem

If a function $g_i(\mathbf{x})$ is convex, then the set $g_i(\mathbf{x}) \leq e_i$ is convex, where e_i is any constant. If functions $g_i(\mathbf{x})$ for $i = 1$ to m are convex, then the set defined by $g_i(\mathbf{x}) \leq e_i$ for $i = 1$ to m is also convex. The set $g_i(\mathbf{x}) \leq e_i$ for $i = 1$ to m is called the intersection of sets defined by the individual constraints $g_i(\mathbf{x}) \leq e_i$. Therefore, intersection of convex sets is a convex set. We can relate convexity of functions and sets by the following theorem:

Theorem 4.9 Convex Functions and Convex Sets Let a set S be defined with constraints of the general optimization problem in Eqs (4.37) to (4.39) as

$$S = \{\mathbf{x} | h_i(\mathbf{x}) = 0, i = 1 \text{ to } p; \quad g_j(\mathbf{x}) \leq 0, j = 1 \text{ to } m\} \quad (4.68)$$

Then S is a convex set if functions g_j are convex and h_i are linear.

The set S of Example 4.36 is convex because it is defined by a convex function. It is important to realize that if we have a nonlinear equality constraint $h_i(\mathbf{x}) = 0$, then the feasible set S is always nonconvex. This can be easily seen from the definition of a convex set. For an equality constraint, the set S is a collection of points lying on the surface $h_i(\mathbf{x}) = 0$. If we take any two points on the surface, the straight line joining them cannot be on the surface, unless it is a plane (linear equality). Therefore, a feasible set defined by any nonlinear equality constraint is always nonconvex. On the contrary, a feasible set defined by a linear equality or inequality is always convex.

If all inequality constraint functions for an optimum design problem are convex, and all equality constraint are linear, then the feasible set S is convex by Theorem 4.9. If the cost function is also convex over, then we have what is known as a *convex programming problem*. Such problems have a very useful property that KKT necessary conditions are also sufficient and any local minimum is also a global minimum.

It is important to note that Theorem 4.9 does not say that the feasible set S cannot be convex if a constraint function $g_i(\mathbf{x})$ fails the convexity check, i.e., it is not an “if and only if” theorem. There are some problems having inequality constraint functions that fail the convexity check, but the feasible set is still convex. Thus, *the condition that $g_i(\mathbf{x})$ be convex for the region $g_i(\mathbf{x}) \leq 0$ to be convex are only sufficient but not necessary*.

Theorem 4.10 Global Minimum If $f(\mathbf{x}^*)$ is a local minimum for a convex function $f(\mathbf{x})$ defined on a convex feasible set S , then it is also a global minimum.

It is important to note that the theorem does not say that \mathbf{x}^* cannot be a global minimum point if functions of the problem fail the convexity test. The point may indeed be a global minimum; however, we cannot claim global optimality using Theorem 4.10. We will have to use some other procedure, such as exhaustive search. Note also that the theorem does not say that the global minimum is unique; i.e., there can be multiple minimum points in the feasible set, all having the same cost function value. The convexity of several problems is checked in Examples 4.39 to 4.41.

EXAMPLE 4.39 Check for Convexity of a Problem

Minimize $f(x_1, x_2) = x_1^3 - x_2^3$ subject to the constraints $x_1 \geq 0, x_2 \leq 0$.

Solution. The constraints actually define the domain for the function $f(\mathbf{x})$ which is the fourth quadrant of a plane (shown in Fig. 4-31). This domain is convex. The Hessian of f is given as

$$\mathbf{H} = \begin{bmatrix} 6x_1 & 0 \\ 0 & -6x_2 \end{bmatrix}$$

The Hessian is positive semidefinite or positive definite over the domain defined by the constraints ($x_1 \geq 0, x_2 \leq 0$). Therefore, the cost function is convex and the problem is convex. Note that if constraints $x_1 \geq 0$ and $x_2 \leq 0$ are not imposed, then the cost function will not be convex for all feasible \mathbf{x} . This can be observed in Fig. 4-31 where several cost function contours are also shown. Thus, the condition of positive semidefiniteness of the Hessian can define the domain for the function over which it is convex.

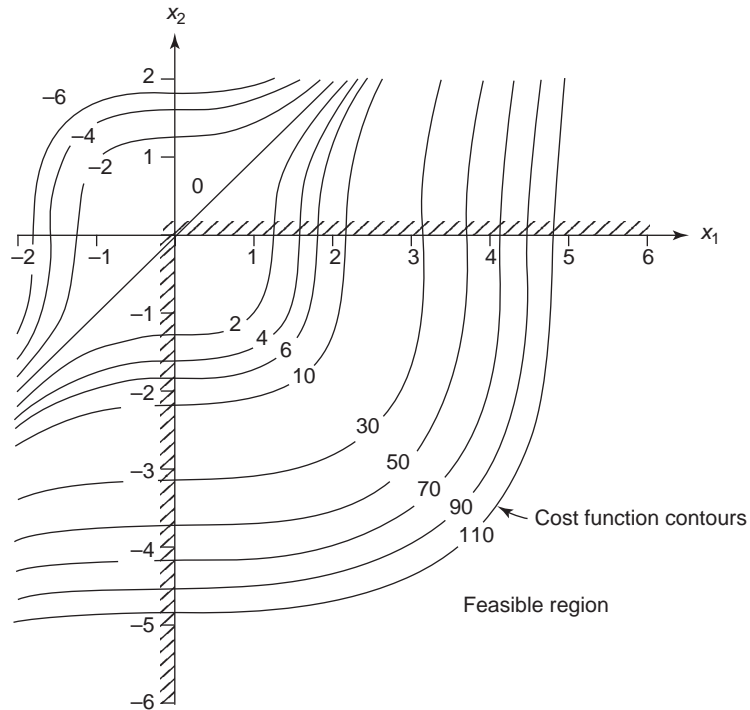


FIGURE 4-31 Graphical representation of Example 4.39.

EXAMPLE 4.40 Check for Convexity of a Problem

Minimize $f(x_1, x_2) = 2x_1 + 3x_2 - x_1^3 - 2x_2^2$ subject to the constraints

$$x_1 + 3x_2 \leq 6, \quad 5x_1 + 2x_2 \leq 10, \quad x_1, x_2 \geq 0$$

Solution. Since all the constraint functions are linear in the variables x_1 and x_2 , the feasible set for the problem is convex. If the cost function f is also convex, then the problem is convex. The Hessian of the cost function is

$$\mathbf{H} = \begin{bmatrix} -6x_1 & 0 \\ 0 & -4 \end{bmatrix}$$

The eigenvalues of \mathbf{H} are $-6x_1$ and -4 . Since the first eigenvalue is nonpositive for $x_1 \geq 0$, and the second eigenvalue is negative, the function is not convex (Theorem 4.8), so the problem cannot be classified as a convex programming problem. Global optimality of a local minimum is not guaranteed. Figure 4-32 shows the feasible set for the problem along with several isocost curves. It is seen that the feasible set is convex but the cost function is not. Thus the problem can have multiple local minima having different values for the cost function.

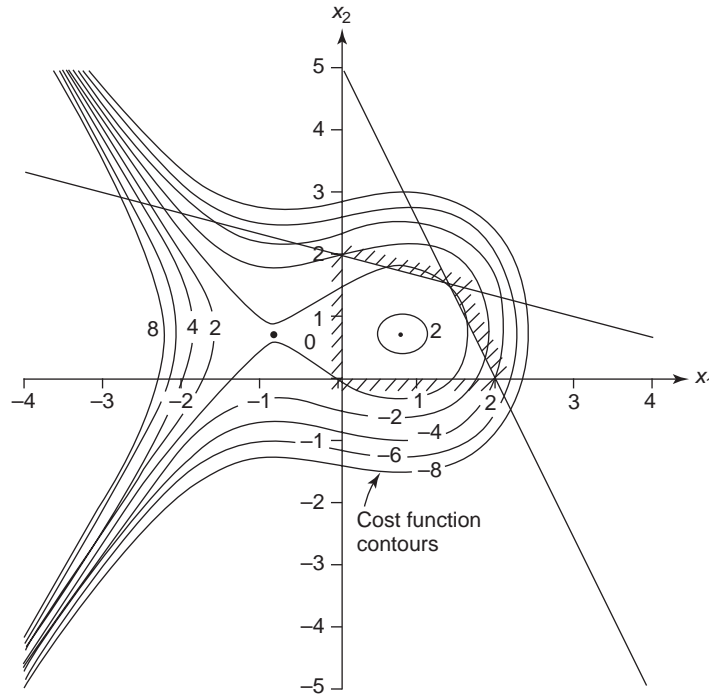


FIGURE 4-32 Graphical representation of Example 4.40.

EXAMPLE 4.41 Check for Convexity of a Problem

Minimize $f(x_1, x_2) = 9x_1^2 - 18x_1x_2 + 13x_2^2 - 4$ subject to $x_1^2 + x_2^2 + 2x_1 \geq 16$.

Solution. To check for convexity of the problem, we need to write the constraint in the standard form as $g(\mathbf{x}) = -x_1^2 - x_2^2 - 2x_1 + 16 \leq 0$. The Hessian of $g(\mathbf{x})$ is

$$\mathbf{H} = \begin{bmatrix} -2 & 0 \\ 0 & -2 \end{bmatrix}$$

Eigenvalues of the Hessian are -2 and -2 . Since, the Hessian is neither positive definite nor positive semidefinite, $g(\mathbf{x})$ is not convex [in fact, the Hessian is negative definite, so $g(\mathbf{x})$ is concave]. Therefore, the problem cannot be classified as a convex programming problem, and global optimality for the solution cannot be guaranteed by Theorem 4.10.

4.6.4 Transformation of a Constraint

A constraint function can be transformed to a different form that is equivalent to the original function, i.e., the constraint boundary and the feasible set for the problem do not change but the form of the function changes. Transformation of a constraint function, however, may affect its convexity check, i.e., *transformed constraint function may fail the convexity check*. Convexity of the feasible set is, however, not affected by the transformation. To illustrate the effect of transformations, let us consider the following inequality constraint:

$$g_1 = \frac{a}{x_1 x_2} - b \leq 0 \quad (a)$$

with $x_1 > 0$, $x_2 > 0$, and a and b as the given positive constants. To check convexity of the constraint, we calculate the Hessian matrix as

$$\nabla^2 g_1 = \frac{2a}{x_1^2 x_2^2} \begin{bmatrix} \frac{x_2}{x_1} & 0.5 \\ 0.5 & \frac{x_1}{x_2} \end{bmatrix} \quad (b)$$

Both eigenvalues as well as the two leading principal minors of the preceding matrix are strictly positive, so the matrix is positive definite and the constraint function g_1 is convex. The feasible set for g_1 is convex.

Now let us transform the constraint by multiplying throughout by $x_1 x_2$ (since $x_1 > 0$, $x_2 > 0$, the sense of the inequality is not changed) to obtain

$$g_2 = a - b x_1 x_2 \leq 0 \quad (c)$$

The constraints g_1 and g_2 are equivalent and will give the same optimum solution for the problem. To check convexity of the constraint function, we calculate the Hessian matrix as

$$\nabla^2 g_2 = \begin{bmatrix} 0 & -b \\ -b & 0 \end{bmatrix} \quad (d)$$

The eigenvalues of the preceding matrix are: $\lambda_1 = -b$ and $\lambda_2 = b$. Therefore, the matrix is indefinite by Theorem 4.2, and by Theorem 4.8, the constraint function g_2 is not convex. Thus, we lose convexity of the constraint function and we cannot claim convexity of the feasible set by Theorem 4.9. Since the problem cannot be shown to be convex, we cannot use results related to convex programming problems.

4.6.5 Sufficient Conditions for Convex Programming Problems

Theorem 4.11 Sufficient Condition for Convex Programming Problem If $f(\mathbf{x})$ is a convex cost function defined on a convex feasible set, then the first-order KKT conditions are necessary as well as sufficient for a global minimum.

Thus, if we can show convexity of a problem, any solution of the necessary conditions will automatically satisfy sufficient conditions (see Example 4.42). In addition, the solution will be a global minimum. Following the procedure of Section 4.4, we consider various cases defined by the switching conditions of Eq. (4.50) until a solution is found. We can stop there as the solution is a *global optimum design*.

EXAMPLE 4.42 Check for Convexity of a Problem

Let us consider Example 4.29 again and check for its convexity. Minimize $f(\mathbf{x}) = (x_1 - 1.5)^2 + (x_2 - 1.5)^2$ subject to $g(\mathbf{x}) = x_1 + x_2 - 2 \leq 0$.

Solution. The KKT necessary conditions give the candidate local minimum as $x_1^* = 1$, $x_2^* = 1$, and $u^* = 1$. The constraint function $g(\mathbf{x})$ is linear, so it is convex. Since the inequality constraint function is convex and there is no equality constraint, the feasible set S is convex. The Hessian matrix for the cost function is

$$\mathbf{H} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Since \mathbf{H} is positive definite everywhere by Theorem 4.2 or Theorem 4.3, the cost function $f(\mathbf{x})$ is strictly convex by Theorem 4.8. Therefore, the problem is convex and the solution $x_1 = 1$, $x_2 = 1$ satisfies sufficiency condition of Theorem 4.11. It is a strict global minimum point for the problem.

The convexity results are summarized in Table 4-3.

TABLE 4-3 Convex Programming Problem—Summary of Results

The problem must be written in the standard form: Minimize $f(\mathbf{x})$ subject to $h_i(\mathbf{x}) = 0$, $g_j(\mathbf{x}) \leq 0$

1. *Convex set.* The geometrical condition that a line joining two points in the set is to be in the set, is an “if and only if” condition for convexity of the set.
2. *Convexity of feasible set S .* All the constraint functions should be convex. *This condition is only sufficient but not necessary;* i.e., functions failing the convexity check may also define convex sets.
 - nonlinear equality constraints always give nonconvex sets
 - linear equalities or inequalities always give convex sets
3. *Convex functions.* A function is convex if and only if its Hessian is at least *positive semidefinite* everywhere.

A function is *strictly convex* if its Hessian is *positive definite* everywhere; however, the *converse is not true*, i.e., a strictly convex function may not have a positive definite Hessian everywhere; thus this condition is only *sufficient* but not necessary.
4. *Form of constraint function.* Changing the form of a constraint function can result in failure of the convexity check for the new constraint or vice versa.
5. *Convex programming problem.* $f(\mathbf{x})$ is convex over the convex feasible set S .
 - KKT first order conditions are necessary as well as sufficient for global minimum
 - Any local minimum point is also a global minimum point

Nonconvex programming problem: If a problem fails convexity checks, it does not imply that there is no global minimum for the problem. It could also have only one local minimum in the feasible set S which would then be a global minimum as well.

4.7 Engineering Design Examples

The procedures described in the previous sections are used to solve two engineering design examples. The problems are formulated, convexity is checked, KKT necessary conditions are written and solved, and the constraint variation sensitivity theorem is illustrated and discussed.

4.7.1 Design of a Wall Bracket

The wall bracket shown in Fig. 4-33 is to be designed to support a load of $W = 1.2$ MN. The material for the bracket should not fail under the action of forces in the bars. These are expressed as the following stress constraints:

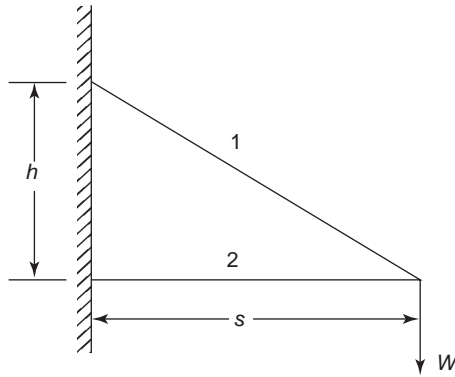


FIGURE 4-33 Wall bracket. $h = 30$ cm, $s = 40$ cm, and $W = 1.2$ MN.

$$\text{Bar 1: } \sigma_1 \leq \sigma_a$$

$$\text{Bar 2: } \sigma_2 \leq \sigma_a$$

where σ_a = allowable stress for the material ($16,000$ N/cm²)

σ_1 = stress in Bar 1 which is given as F_1/A_1 , N/cm²

σ_2 = stress in Bar 2 which is given as F_2/A_2 , N/cm²

A_1 = cross-sectional area of Bar 1 (cm²)

A_2 = cross-sectional area of Bar 2 (cm²)

F_1 = force due to load W in Bar 1 (N)

F_2 = force due to load W in Bar 2 (N)

Total volume of the bracket is to be minimized.

Problem Formulation The cross-sectional areas A_1 and A_2 are the two design variables and the cost function for the problem is the volume, which is given as

$$f(A_1, A_2) = l_1 A_1 + l_2 A_2, \text{ cm}^3 \quad (\text{a})$$

where $l_1 = \sqrt{30^2 + 40^2} = 50$ cm is the length of member 1 and $l_2 = 40$ cm is the length of member 2. To write the stress constraints, we need forces in the members which are obtained using static equilibrium analysis as follows: $F_1 = 2.0 \times 10^6$ N, $F_2 = 1.6 \times 10^6$ N. Therefore, stress constraints are given as

$$g_1 = \frac{(2.0 \times 10^6)}{A_1} - 16000 \leq 0 \quad (\text{b})$$

$$g_2 = \frac{(1.6 \times 10^6)}{A_2} - 16000 \leq 0 \quad (\text{c})$$

The cross-sectional areas must both be nonnegative:

$$g_3 = -A_1 \leq 0, \quad g_4 = -A_2 \leq 0 \quad (\text{d})$$

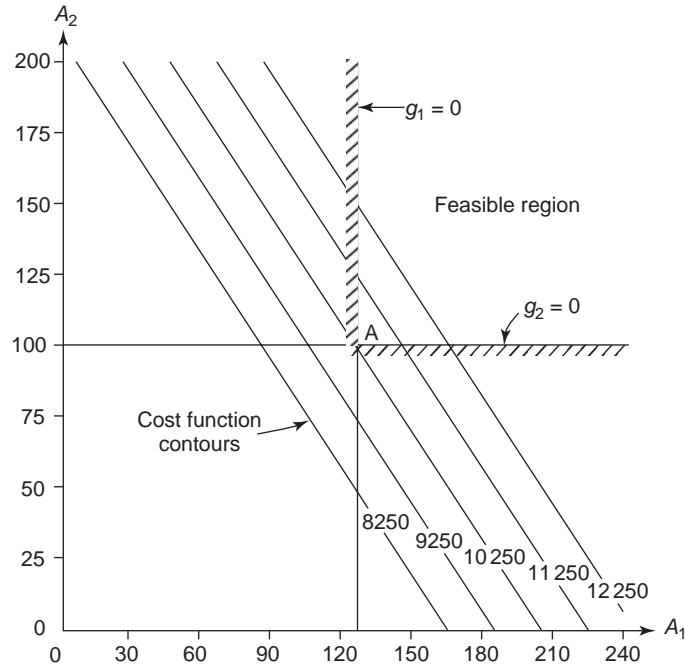


FIGURE 4-34 Graphical solution for the wall bracket problem.

Constraints for the problem are plotted in Fig. 4-34, and the feasible region is identified. A few cost function contours are also shown. It can be seen that the optimum solution is at point A with $A_1^* = 125 \text{ cm}^2$, $A_2^* = 100 \text{ cm}^2$, and $f = 10,250 \text{ cm}^3$.

Convexity Since the cost function of Eq. (a) is linear in terms of design variables, it is convex. The Hessian matrix for the constraint g_1 is

$$\nabla^2 g_1 = \begin{bmatrix} \frac{(4.0 \times 10^6)}{A_1^3} & 0 \\ 0 & 0 \end{bmatrix}$$

which is a positive semidefinite matrix for $A_1 > 0$, so g_1 is convex. Similarly, g_2 is convex, and since g_3 and g_4 are linear, they are convex. Thus the problem is *convex*, and KKT necessary conditions are also *sufficient* and any design satisfying the KKT conditions is a global minimum.

KKT Necessary Conditions To use the KKT conditions, we introduce slack variables into the constraints and define the Lagrange function of Eq. (4.46a) for the problem as

$$L = (l_1 A_1 + l_2 A_2) + u_1 \left[\frac{(2.0 \times 10^6)}{A_1} - 16000 + s_1^2 \right] + u_2 \left[\frac{(1.6 \times 10^6)}{A_2} - 16000 + s_2^2 \right] + u_3 (-A_1 + s_3^2) + u_4 (-A_2 + s_4^2) \quad (e)$$

TABLE 4-4 Definition of Karush-Kuhn-Tucker Cases with Four Inequalities

No.	Case	Active Constraints
1	$u_1 = 0, u_2 = 0, u_3 = 0, u_4 = 0$	No inequality active
2	$s_1 = 0, u_2 = 0, u_3 = 0, u_4 = 0$	One inequality active at a time
3	$u_1 = 0, s_2 = 0, u_3 = 0, u_4 = 0$	
4	$u_1 = 0, u_2 = 0, s_3 = 0, u_4 = 0$	
5	$u_1 = 0, u_2 = 0, u_3 = 0, s_4 = 0$	
6	$s_1 = 0, s_2 = 0, u_3 = 0, u_4 = 0$	Two inequalities active at a time
7	$u_1 = 0, s_2 = 0, s_3 = 0, u_4 = 0$	
8	$u_1 = 0, u_2 = 0, s_3 = 0, s_4 = 0$	
9	$s_1 = 0, u_2 = 0, u_3 = 0, s_4 = 0$	
10	$s_1 = 0, u_2 = 0, s_3 = 0, u_4 = 0$	
11	$u_1 = 0, s_2 = 0, u_3 = 0, s_4 = 0$	
12	$s_1 = 0, s_2 = 0, s_3 = 0, u_4 = 0$	Three inequalities active at a time
13	$u_1 = 0, s_2 = 0, s_3 = 0, s_4 = 0$	
14	$s_1 = 0, u_2 = 0, s_3 = 0, s_4 = 0$	
15	$s_1 = 0, s_2 = 0, u_3 = 0, s_4 = 0$	
16	$s_1 = 0, s_2 = 0, s_3 = 0, s_4 = 0$	Four inequalities active at a time

The necessary conditions become

$$\frac{\partial L}{\partial A_1} = l_1 - u_1 \frac{(2.0 \times 10^6)}{A_1^2} - u_3 = 0 \quad (f)$$

$$\frac{\partial L}{\partial A_2} = l_2 - u_2 \frac{(1.6 \times 10^6)}{A_2^2} - u_4 = 0 \quad (g)$$

$$u_i s_i = 0, u_i \geq 0, g_i + s_i^2 = 0, s_i^2 \geq 0; \quad i = 1 \text{ to } 4 \quad (h)$$

The switching conditions in Eq. (h) give 16 solution cases. These case can be identified using a systematic procedure as shown in Table 4-4. Note that any case that requires $s_3 = 0$ (i.e., $g_3 = 0$) makes the area $A_1 = 0$. For such a case the constraint g_1 of Eq. (b) is violated, so it does not give a candidate solution. Similarly, $s_4 = 0$ makes $A_2 = 0$, which violates the constraint of Eq. (c). In addition, A_1 and A_2 cannot be negative because the corresponding solution has no physical meaning. Therefore, all the cases requiring either $s_3 = 0$ and/or $s_4 = 0$ do not give any candidate solution. These cases need not be considered any further. This leaves only cases 1 to 3 and 6 for further consideration, and we solve them as follows (any case giving $A_1 < 0$ or $A_2 < 0$ will also be discarded).

Case 1: $u_1 = 0, u_2 = 0, u_3 = 0, u_4 = 0$. This case gives $l_1 = 0$ and $l_2 = 0$ in Eqs. (f) and (g) which is *not acceptable*.

Case 2: $s_1 = 0, u_2 = 0, u_3 = 0, u_4 = 0$. This gives $l_1 = 0$ in Eq. (f) which is *not acceptable*.

Case 3: $u_1 = 0, s_2 = 0, u_3 = 0, u_4 = 0$. This gives $l_2 = 0$ in Eq. (g) which is *not acceptable*.

Case 6: $s_1 = 0, s_2 = 0, u_3 = 0, u_4 = 0$. Equations (b) and (c) give $A_1^* = 125 \text{ cm}^2, A_2^* = 100 \text{ cm}^2$. Equations (f) and (g) give the Lagrange multipliers as $u_1 = 0.391$ and $u_2 = 0.25$ and since both are nonnegative, all the KKT conditions are satisfied. The cost function at optimum is obtained as $f^* = 50(125) + 40(100)$ or $f^* = 10,250 \text{ cm}^3$. The

gradients of active constraints are $(-(2.0 \times 10^6)/A_1^2, 0)$ and $(0, -(1.0 \times 10^6)/A_2^2)$. These vectors are linearly independent, and so the minimum point is a regular point of the feasible set.

Sensitivity Analysis If the allowable stress changes to $16,500 \text{ N/cm}^2$ from $16,000 \text{ N/cm}^2$, we need to know how the cost function will change. Using Eq. (4.56) we get the change in the cost function as $\delta f^* = -u_1 e_1 - u_2 e_2$, where $e_1 = e_2 = 16,500 - 16,000 = 500 \text{ N/cm}^2$. Therefore, the change in the cost function is $\delta f^* = -0.391(500) - 0.25(500) = -320.5 \text{ cm}^3$. Thus the volume of the bracket will reduce by 320.5 cm^3 .

4.7.2 Design of a Rectangular Beam

In Section 3.8, a rectangular beam design problem is formulated and solved graphically. We will solve the same problem using the KKT necessary conditions. The problem is formulated as follows. Find b and d to minimize

$$f(b, d) = bd \quad (\text{a})$$

subject to the inequality constraints

$$g_1 = \frac{(2.40 \times 10^8)}{bd^2} - 10 \leq 0 \quad (\text{b})$$

$$g_2 = \frac{(2.25 \times 10^5)}{bd} - 2 \leq 0 \quad (\text{c})$$

$$g_3 = d - 2b \leq 0 \quad (\text{d})$$

$$g_4 = -b \leq 0, \quad g_5 = -d \leq 0 \quad (\text{e})$$

Convexity Constraints $g_3, g_4,$ and g_5 are linear in terms of b and d , and are therefore convex. The Hessian for the constraint g_1 is given as

$$\nabla^2 g_1 = \frac{(4.80 \times 10^8)}{b^3 d^4} \begin{bmatrix} d^2 & bd \\ bd & 3b^2 \end{bmatrix}$$

Since this matrix is positive definite for $b > 0$ and $d > 0$, g_1 is a strictly convex function. The Hessian for the constraint g_2 is given as

$$\nabla^2 g_2 = \frac{(2.25 \times 10^5)}{b^3 d^3} \begin{bmatrix} 2d^2 & bd \\ bd & 2b^2 \end{bmatrix}$$

Since this matrix is positive definite, the constraint g_2 is also strictly convex. Since all the constraints of the problem are convex, the feasible set is convex.

It is interesting to note that *constraints* g_1 and g_2 can be *transformed* as (since $b > 0$ and $d > 0$, the sense of inequality is not changed):

$$\bar{g}_1 = (2.40 \times 10^8) - 10bd^2 \leq 0 \quad (\text{f})$$

$$\bar{g}_2 = (2.25 \times 10^5) - 2bd \leq 0 \quad (\text{g})$$

Hessians of the functions \bar{g}_1 and \bar{g}_2 are given as

$$\nabla^2 \bar{g}_1 = \begin{bmatrix} 0 & -20d \\ -20d & -20b \end{bmatrix} \quad \nabla^2 \bar{g}_2 = \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix}$$

Both of the preceding matrices are not positive semidefinite. Therefore, the constraint functions \bar{g}_1 and \bar{g}_2 given in Eqs. (f) and (g) are not convex. *This goes to show that convexity of a function can be lost if it is transformed to another form.* This is an important observation, and it shows that we should be careful in transformation of constraint functions. Note, however, that transformation of constraints does not change the optimum solution. It does change the values of the Lagrange multipliers for the constraints, however, as discussed in Section 4.5.

In order to check convexity of the cost function, we write its Hessian as

$$\nabla^2 f = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (\text{h})$$

This matrix is indefinite, so the cost function is nonconvex. The problem fails the convexity check of Theorem 4.9, and we cannot guarantee global optimality of the solution by Theorem 4.10. *Note that this does not say that a local minimum cannot be a global minimum.* It may still be a global minimum, but cannot be guaranteed by Theorem 4.10.

KKT Necessary Conditions To use the KKT conditions, we introduce slack variables into the constraints and define the Lagrange function for the problem as

$$L = bd + u_1 \left(\frac{(2.40 \times 10^8)}{bd^2} - 10 + s_1^2 \right) + u_2 \left(\frac{(2.25 \times 10^5)}{bd} - 2 + s_2^2 \right) + u_3 (d - 2b + s_3^2) \\ + u_4 (-b + s_4^2) + u_5 (-d + s_5^2)$$

The necessary conditions give

$$\frac{\partial L}{\partial b} = d + u_1 \frac{(-2.40 \times 10^8)}{b^2 d^2} + u_2 \frac{(-2.25 \times 10^5)}{b^2 d} - 2u_3 - u_4 = 0 \quad (\text{i})$$

$$\frac{\partial L}{\partial d} = b + u_1 \frac{(-4.80 \times 10^8)}{bd^3} + u_2 \frac{(-2.25 \times 10^5)}{bd^2} + u_3 - u_5 = 0 \quad (\text{j})$$

$$u_i s_i = 0, u_i \geq 0, g_i + s_i^2 = 0; s_i^2 \geq 0; \quad i = 1 \text{ to } 5 \quad (\text{k})$$

The switching conditions in Eq. (k) give 32 cases for the necessary conditions. However, note that the cases requiring either $s_4 = 0$ or $s_5 = 0$, or both as zero, do not give any candidate optimum points because they violate the constraint of either Eqs. (b) and (c) or Eq. (d). Therefore, these cases shall not be considered, which can be done by setting $u_4 = 0$ and $u_5 = 0$ in the remaining cases. This leaves the following eight cases for further consideration:

1. $u_1 = 0, u_2 = 0, u_3 = 0, u_4 = 0, u_5 = 0$
2. $u_1 = 0, u_2 = 0, s_3 = 0, u_4 = 0, u_5 = 0$
3. $u_1 = 0, s_2 = 0, u_3 = 0, u_4 = 0, u_5 = 0$
4. $s_1 = 0, u_2 = 0, u_3 = 0, u_4 = 0, u_5 = 0$
5. $u_1 = 0, s_2 = 0, s_3 = 0, u_4 = 0, u_5 = 0$
6. $s_1 = 0, s_2 = 0, u_3 = 0, u_4 = 0, u_5 = 0$
7. $s_1 = 0, u_2 = 0, s_3 = 0, u_4 = 0, u_5 = 0$
8. $s_1 = 0, s_2 = 0, s_3 = 0, u_4 = 0, u_5 = 0$

We consider each case at a time and solve for the candidate optimum points. Note that any solution having $b < 0$ or $d < 0$ violates constraints g_4 or g_5 and shall be discarded.

Case 1: $u_1 = 0, u_2 = 0, u_3 = 0, u_4 = 0, u_5 = 0$. This case gives $d = 0, b = 0$ in Eqs. (i) and (j). Therefore, this case does not give a solution.

Case 2: $u_1 = 0, u_2 = 0, s_3 = 0, u_4 = 0, u_5 = 0$. Equation (d) gives $d = 2b$. Equations (i) and (j) give $d - 2u_3 = 0$ and $d + u_3 = 0$. These three equations give $b = 0$ and $d = 0$, which is not feasible.

Case 3: $u_1 = 0, s_2 = 0, u_3 = 0, u_4 = 0, u_5 = 0$. Equations (i), (j), and (c) give

$$d - u_2 \frac{(2.25 \times 10^5)}{b^2 d} = 0$$

$$b - u_2 \frac{(2.25 \times 10^5)}{bd^2} = 0$$

$$\frac{(2.25 \times 10^5)}{bd} - 2 = 0$$

These equations give a solution as $u_2 = (5.625 \times 10^4)$ and $bd = (1.125 \times 10^5)$. Since $u_2 > 0$, this is a valid solution. Actually, there is a family of solutions given by $bd = (1.125 \times 10^5)$; for any $d > 0$, b can be found from this equation. However, there must be some limits on the values of b and d for which this family of solutions is valid. These ranges are provided by requiring $s_1^2 \geq 0$ and $s_3^2 \geq 0$, or $g_1 \leq 0$ and $g_3 \leq 0$.

Substituting $b = (1.125 \times 10^5)/d$ into g_1 (Eq. b),

$$\frac{(2.40 \times 10^8)}{(1.125 \times 10^5)d} - 10 \leq 0 \quad \text{or} \quad d \geq 213.33 \text{ mm} \quad (l)$$

Substituting $b = (1.125 \times 10^5)/d$ into g_3 (Eq. d),

$$d - \frac{(2.25 \times 10^5)}{bd} \leq 0; \quad \text{or} \quad d \leq 474.34 \text{ mm} \quad (m)$$

This gives limits on the depth d . We can find limits on the width b by substituting Eqs. (l) and (m) into $bd = (1.125 \times 10^5)$:

$$d \geq 213.33, \quad b \leq 527.34$$

$$d \leq 474.33, \quad b \geq 237.17$$

Therefore, for this case the possible solutions are

$$237.17 \leq b \leq 527.34 \text{ mm}; \quad 213.33 \leq d \leq 474.33 \text{ mm}$$

$$bd = (1.125 \times 10^5) \text{ mm}^2$$

Case 4: $s_1 = 0, u_2 = 0, u_3 = 0, u_4 = 0, u_5 = 0$. Equations (i) and (j) reduce to

$$d - \frac{(2.40 \times 10^8)}{b^2 d^2} = 0; \quad \text{or} \quad b^2 d^3 = (2.40 \times 10^8)$$

$$b - \frac{(4.80 \times 10^8)}{bd^3} = 0; \quad \text{or} \quad b^2 d^3 = (4.80 \times 10^8)$$

Since the previous two equations are inconsistent, there is no solution for this case.

Case 5: $u_1 = 0, s_2 = 0, s_3 = 0, u_4 = 0, u_5 = 0$. Equations (c) and (d) can be solved for b and d , e.g., substituting $b = 2d$ from Eq. (d) into Eq. (c), we get $b = 237.17$ mm. Therefore, $d = 2(237.17) = 474.34$ mm. We can calculate u_2 and u_3 from Eqs. (i) and (j) as $u_2 = (5.625 \times 10^4)$, $u_3 = 0$. Substituting values of b and d into Eq. (b), we get $g_1 = -5.5 < 0$, so the constraint is satisfied (i.e., $s_1^2 > 0$). It can be verified that the gradients of g_2 and g_3 at the candidate point are linearly independent, and so the regularity condition is satisfied. Since all the necessary conditions are satisfied, this is a valid solution. The constraint sensitivity Theorem 4.7 and Eq. (4.54) tell us that since $u_3 = 0$, we can move away from that constraint toward the feasible region without affecting the optimum cost function value. This can also be observed from Fig. 3-11 where the graphical solution for the problem is given. In the figure, point B represents the solution for this case. We can leave point B toward point A and remain on the constraint $g_2 = 0$ for optimum designs.

Case 6: $s_1 = 0, s_2 = 0, u_3 = 0, u_4 = 0, u_5 = 0$. Equations (b) and (c) can be solved for the band d as $b = 527.34$ mm and $d = 213.33$ mm. We can solve for u_1 and u_2 from Eqs. (i) and (j) as $u_1 = 0$ and $u_2 = (5.625 \times 10^4)$. Substituting values of b and d into Eq. (d), we get $g_3 = -841.35 < 0$, so the constraint is satisfied (i.e., $s_3^2 \geq 0$). It can also be verified that the point also satisfies the regularity condition. Since all the KKT conditions are satisfied, this is a valid solution. This solution is quite similar to the one for case 5. The solution corresponds to point A in Fig. 3-11. If we leave constraint $g_1 = 0$ (point A) and remain on the curve A-B, we obtain other optimum designs near the point A.

Case 7: $s_1 = 0, u_2 = 0, s_3 = 0, u_4 = 0, u_5 = 0$. Equations (b) and (d) can be solved as $b = 181.71$ mm and $d = 363.42$ mm. Equations (i) and (j) give the Lagrange multipliers $u_1 = 4402.35$ and $u_3 = -60.57$. Since $u_3 < 0$, this case does not give a valid solution.

Case 8: $s_1 = 0, s_2 = 0, s_3 = 0, u_4 = 0, u_5 = 0$. This case gives three equations in two unknowns (over-determined system), which has no solution.

Sensitivity Analysis It should be observed that none of the candidate minimum points (Points A and B and curve A-B in Fig. 3-11) satisfies the sufficiency conditions presented in the next chapter. Therefore, *the existence of partial derivatives of the cost function with respect to the right side parameters of Eq. (4.54) is not guaranteed by Theorem 4.7*. However, since we have a graphical solution for the problem in Fig. 3-11, we can check what happens if we do use the sensitivity theorem.

For Point A in Fig. 3-11 (case 6), constraints g_1 and g_2 are active, $b = 527.34$ mm, $d = 213.33$ mm, $u_1 = 0$, and $u_2 = (5.625 \times 10^4)$. Since $u_1 = 0$, Eq. (4.54) gives $\partial f / \partial e_1 = 0$. This means any small change in the constraint limit does not change the optimum cost function value. This is true, which can be observed from Fig. 3-11. The optimum point is changed but constraint g_1 remains active; i.e., $bd = (1.125 \times 10^5)$ must be satisfied. Any change in g_2 moves the constraint parallel to itself, changing the optimum solution (design variables and the cost function). Since $u_2 = (5.625 \times 10^4)$, Eq. (4.54) gives $\partial f / \partial e_2 = (-5.625 \times 10^4)$. It can be verified that the sensitivity coefficient predicts correct changes in the cost function.

It can be verified that the other two solution cases (3 and 5) also give correct values for the sensitivity coefficients.

Exercises for Chapter 4

Section 4.2 Review of Some Basic Calculus Concepts

4.1 Answer True or False.

1. A function can have several local minimum points in a small neighborhood of \mathbf{x}^* .
2. A function cannot have more than one global minimum point.
3. The value of the function having global minimum at several points must be the same.
4. A function defined on an open set cannot have a global minimum.
5. The gradient of a function $f(\mathbf{x})$ at a point is normal to the surface defined by the level surface $f(\mathbf{x}) = \text{constant}$.
6. Gradient of a function at a point gives a local direction of maximum decrease in the function.
7. The Hessian matrix of a continuously differentiable function can be asymmetric.
8. The Hessian matrix for a function is calculated using only the first derivatives of the function.
9. Taylor series expansion for a function at a point uses the function value and its derivatives.
10. Taylor series expansion can be written at a point where the function is discontinuous.
11. Taylor series expansion of a complicated function replaces it with a polynomial function at the point.
12. Linear Taylor series expansion of a complicated function at a point is only a good local approximation for the function.
13. A quadratic form can have first-order terms in the variables.
14. For a given \mathbf{x} , the quadratic form defines a vector.
15. Every quadratic form has a symmetric matrix associated with it.
16. A symmetric matrix is positive definite if its eigenvalues are nonnegative.
17. A matrix is positive semidefinite if some of its eigenvalues are negative and others are nonnegative.
18. All eigenvalues of a negative definite matrix are strictly negative.
19. The quadratic form appears as one of the terms in Taylor's expansion of a function.
20. A positive definite quadratic form must have positive value for any $\mathbf{x} \neq \mathbf{0}$.

Write the Taylor series expansion for the following functions up to quadratic terms.

- 4.2 $\cos x$ about the point $x^* = \pi/4$
- 4.3 $\cos x$ about the point $x^* = \pi/3$
- 4.4 $\sin x$ about the point $x^* = \pi/6$
- 4.5 $\sin x$ about the point $x^* = \pi/4$
- 4.6 e^x about the point $x^* = 0$

4.7 e^x about the point $x^* = 2$

4.8 $f(x_1, x_2) = 10x_1^4 - 20x_1^2x_2 + 10x_2^2 + x_1^2 - 2x_1 + 5$ about the point $(1, 1)$. Compare approximate and exact values of the function at the point $(1.2, 0.8)$.

Determine the nature of the following quadratic forms.

4.9 $F(\mathbf{x}) = x_1^2 + 4x_1^2x_2 + 2x_1x_3 - 7x_2^2 - 6x_2x_3 + 5x_3^2$

4.10 $F(\mathbf{x}) = 2x_1^2 + 2x_2^2 - 5x_1x_2$

4.11 $F(\mathbf{x}) = x_1^2 + x_2^2 + 3x_1x_2$

4.12 $F(\mathbf{x}) = 3x_1^2 + x_2^2 - x_1x_2$

4.13 $F(\mathbf{x}) = x_1^2 - x_2^2 + 4x_1x_2$

4.14 $F(\mathbf{x}) = x_1^2 - x_2^2 + x_3^2 - 2x_2x_3$

4.15 $F(\mathbf{x}) = x_1^2 - 2x_1x_2 + 2x_2^2$

4.16 $F(\mathbf{x}) = x_1^2 - x_1x_2 - x_2^2$

4.17 $F(\mathbf{x}) = x_1^2 + 2x_1x_3 - 2x_2^2 + 4x_3^2 - 2x_2x_3$

4.18 $F(\mathbf{x}) = 2x_1^2 + x_1x_2 + 2x_2^2 + 3x_3^2 - 2x_1x_3$

4.19 $F(\mathbf{x}) = x_1^2 + 2x_2x_3 + x_2^2 + 4x_3^2$

4.20 $F(\mathbf{x}) = 4x_1^2 + 2x_1x_3 - x_2^2 + 4x_3^2$

Section 4.3 Unconstrained Optimum Design Problems

4.21 *Answer True or False.*

1. If the first-order necessary condition at a point is satisfied for an unconstrained problem, it can be a local maximum point for the function.
2. A point satisfying first-order necessary conditions for an unconstrained function may not be a local minimum point.
3. A function can have a negative value at its maximum point.
4. If a constant is added to a function, the location of its minimum point is changed.
5. If a function is multiplied by a positive constant, the location of the function's minimum point is unchanged.
6. If curvature of an unconstrained function of a single variable at the point x^* is zero, then it is a local maximum point for the function.
7. The curvature of an unconstrained function of a single variable at its local minimum point is negative.
8. The Hessian of an unconstrained function at its local minimum point must be positive semidefinite.
9. The Hessian of an unconstrained function at its minimum point is negative definite.
10. If the Hessian of an unconstrained function is indefinite at a candidate point, the point may be a local maximum or minimum.

Find stationary points for the following functions (use a numerical method such as the Newton-Raphson method in Appendix C, or a software package like Excel, MATLAB, and Mathematica, if needed). Also determine the local minimum, local maximum, and inflection points for the functions (inflection points are those stationary points that are neither minimum nor maximum).

$$4.22 \quad f(x_1, x_2) = 3x_1^2 + 2x_1x_2 + 2x_2^2 + 7$$

$$4.23 \quad f(x_1, x_2) = x_1^2 + 4x_1x_2 + x_2^2 + 3$$

$$4.24 \quad f(x_1, x_2) = x_1^3 + 12x_1x_2^2 + 2x_2^2 + 5x_1^2 + 3x_2$$

$$4.25 \quad f(x_1, x_2) = 5x_1 - \frac{1}{16}x_1^2x_2 + \frac{1}{4x_1}x_2^2$$

$$4.26 \quad f(x) = \cos x$$

$$4.27 \quad f(x_1, x_2) = x_1^2 + x_1x_2 + x_2^2$$

$$4.28 \quad f(x) = x^2e^{-x}$$

$$4.29 \quad f(x_1, x_2) = x_1 - \frac{10}{x_1x_2} + 5x_2$$

$$4.30 \quad f(x_1, x_2) = x_1^2 - 2x_1 + 4x_2^2 - 8x_2 + 6$$

$$4.31 \quad f(x_1, x_2) = 3x_1^2 - 2x_1x_2 + 5x_2^2 + 8x_2$$

4.32 The annual operating cost U for an electrical line system is given by the following expression

$$U = \frac{(21.9E+07)}{V^2C} + (3.9E+06)C + (1.0E+03)V$$

where V = line voltage in kilovolts and C = line conductance in mhos. Find stationary points for the function, and determine V and C to minimize the operating cost.

$$4.33 \quad f(x_1, x_2) = x_1^2 + 2x_2^2 - 4x_1 - 2x_1x_2$$

$$4.34 \quad f(x_1, x_2) = 12x_1^2 + 22x_2^2 - 1.5x_1 - x_2$$

$$4.35 \quad f(x_1, x_2) = 7x_1^2 + 12x_2^2 - x_1$$

$$4.36 \quad f(x_1, x_2) = 12x_1^2 + 21x_2^2 - x_2$$

$$4.37 \quad f(x_1, x_2) = 25x_1^2 + 20x_2^2 - 2x_1 - x_2$$

$$4.38 \quad f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$$

$$4.39 \quad f(x_1, x_2) = 8x_1^2 + 8x_2^2 - 80\sqrt{x_1^2 + x_2^2} - 20x_2 + 100 \\ - 80\sqrt{x_1^2 + x_2^2} + 20x_2 + 100 - 5x_1 - 5x_2$$

$$4.40 \quad f(x_1, x_2) = 9x_1^2 + 9x_2^2 - 100\sqrt{x_1^2 + x_2^2} - 20x_2 + 100 \\ - 64\sqrt{x_1^2 + x_2^2} + 16x_2 + 64 - 5x_1 - 41x_2$$

$$4.41 \quad f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

$$4.42 \quad f(x_1, x_2, x_3, x_4) = (x_1 - 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

Section 4.4 Constrained Optimum Design Problems

4.43 Answer True or False.

1. A regular point of the feasible region is defined as a point where the cost function gradient is independent of the gradients of active constraints.

2. A point satisfying KKT conditions for a general optimum design problem can be a local max-point for the cost function.
3. At the optimum point, the number of active independent constraints is always more than the number of design variables.
4. In the general optimum design problem formulation, the number of independent equality constraints must be " \leq " to the number of design variables.
5. In the general optimum design problem formulation, the number of inequality constraints cannot exceed the number of design variables.
6. At the optimum point, Lagrange multipliers for the " \leq type" inequality constraints must be nonnegative.
7. At the optimum point, the Lagrange multiplier for a " \leq type" constraint can be zero.
8. While solving an optimum design problem by KKT conditions, each case defined by the switching conditions can have multiple solutions.
9. In optimum design problem formulation, " \geq type" constraints cannot be treated.
10. Optimum design points for constrained optimization problems give stationary value to the Lagrange function with respect to design variables.
11. Optimum design points having at least one active constraint give stationary value to the cost function.
12. At a constrained optimum design point that is regular, the cost function gradient is linearly dependent on the gradients of the active constraint functions.
13. If a slack variable has zero value at the optimum, the inequality constraint is inactive.
14. Gradients of inequality constraints that are active at the optimum point must be zero.
15. Design problems with equality constraints have the gradient of the cost function as zero at the optimum point.

Find points satisfying KKT necessary conditions for the following problems; check if they are optimum points using the graphical method (if possible).

- 4.44 Minimize $f(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8x_1$
subject to $x_1 + x_2 = 4$
- 4.45 Minimize $f(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8x_1$
subject to $x_1 + x_2 = 4$
- 4.46 Minimize $f(x_1, x_2) = (x_1 - 2)^2 + (x_2 + 1)^2$
subject to $2x_1 + 3x_2 - 4 = 0$
- 4.47 Minimize $f(x_1, x_2) = 4x_1^2 + 9x_2^2 + 6x_2 - 4x_1 + 13$
subject to $x_1 - 3x_2 + 3 = 0$
- 4.48 Minimize $f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 + 2)^2 + (x_3 - 2)^2$
subject to $2x_1 + 3x_2 - 1 = 0$
 $x_1 + x_2 + 2x_3 - 4 = 0$
- 4.49 Minimize $f(x_1, x_2) = 9x_1^2 + 18x_1x_2 + 13x_2^2 - 4$
subject to $x_1^2 + x_2^2 + 2x_1 = 16$
- 4.50 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
subject to $x_1 + x_2 - 4 = 0$

- 4.51 Consider the following problem with equality constraints:
 Minimize $(x_1 - 1)^2 + (x_2 - 1)^2$
 subject to $x_1 + x_2 - 4 = 0$
 $x_1 - x_2 - 2 = 0$
1. Is it a valid optimization problem? Explain.
 2. Explain how you would solve the problem? Are necessary conditions needed to find the optimum solution?
- 4.52 Minimize $f(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8$
 subject to $x_1 + x_2 = 4$
- 4.53 Maximize $F(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8$
 subject to $x_1 + x_2 = 4$
- 4.54 Maximize $F(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8$
 subject to $x_1 + x_2 \leq 4$
- 4.55 Minimize $f(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8$
 subject to $x_1 + x_2 \leq 4$
- 4.56 Maximize $F(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8x_1$
 subject to $x_1 + x_2 \leq 4$
- 4.57 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
 subject to $x_1 + x_2 \geq 4$
 $x_1 - x_2 - 2 = 0$
- 4.58 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
 subject to $x_1 + x_2 = 4$
 $x_1 - x_2 - 2 \geq 0$
- 4.59 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
 subject to $x_1 + x_2 \geq 4$
 $x_1 - x_2 \geq 2$
- 4.60 Minimize $f(x, y) = (x - 4)^2 + (y - 6)^2$
 subject to $12 \geq x + y$
 $x \geq 6, y \geq 0$
- 4.61 Minimize $f(x_1, x_2) = 2x_1 + 3x_2 - x_1^3 - 2x_2^2$
 subject to $x_1 + 3x_2 \leq 6$
 $5x_1 + 2x_2 \leq 10$
 $x_1, x_2 \geq 0$
- 4.62 Minimize $f(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8x_1$
 subject to $x_1 + x_2 \leq 4$
- 4.63 Minimize $f(x_1, x_2) = x_1^2 + x_2^2 - 4x_1 - 2x_2 + 6$
 subject to $x_1 + x_2 \geq 4$
- 4.64 Minimize $f(x_1, x_2) = 2x_1^2 - 6x_1x_2 + 9x_2^2 - 18x_1 + 9x_2$
 subject to $x_1 + 2x_2 \leq 10$
 $4x_1 - 3x_2 \leq 20; x_i \geq 0; i = 1, 2$
- 4.65 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
 subject to $x_1 + x_2 - 4 \leq 0$

- 4.66 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
subject to $x_1 + x_2 - 4 \leq 0$
 $x_1 - x_2 - 2 \geq 0$
- 4.67 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
subject to $x_1 + x_2 - 4 \leq 0$
 $2 - x_1 \leq 0$
- 4.68 Minimize $f(x_1, x_2) = 9x_1^2 - 18x_1x_2 + 13x_2^2 - 4$
subject to $x_1^2 + x_2^2 + 2x_1 \geq 16$
- 4.69 Minimize $f(x_1, x_2) = (x_1 - 3)^2 + (x_2 - 3)^2$
subject to $x_1 + x_2 \leq 4$
 $x_1 - 3x_2 = 1$
- 4.70 Minimize $f(x_1, x_2) = x_1^3 - 16x_1 + 2x_2 - 3x_2^2$
subject to $x_1 + x_2 \leq 3$
- 4.71 Minimize $f(x_1, x_2) = 3x_1^2 - 2x_1x_2 + 5x_2^2 + 8x_2$
subject to $x_1^2 - x_2^2 + 8x_2 \leq 16$
- 4.72 Minimize $f(x, y) = (x - 4)^2 + (y - 6)^2$
subject to $x + y \leq 12$
 $x \leq 6$
 $x, y \geq 0$
- 4.73 Minimize $f(x, y) = (x - 8)^2 + (y - 8)^2$
subject to $x + y \leq 12$
 $x \leq 6$
 $x, y \geq 0$
- 4.74 Maximize $F(x, y) = (x - 4)^2 + (y - 6)^2$
subject to $x + y \leq 12$
 $6 \geq x$
 $x, y \geq 0$
- 4.75 Maximize $F(r, t) = (r - 8)^2 + (t - 8)^2$
subject to $10 \geq r + t$
 $t \leq 5$
 $r, t \geq 0$
- 4.76 Maximize $F(r, t) = (r - 3)^2 + (t - 2)^2$
subject to $10 \geq r + t$
 $t \leq 5$
 $r, t \geq 0$
- 4.77 Maximize $F(r, t) = (r - 8)^2 + (t - 8)^2$
subject to $r + t \leq 10$
 $t \geq 0$
 $r \leq 0$
- 4.78 Maximize $F(r, t) = (r - 3)^2 + (t - 2)^2$
subject to $10 \geq r + t$
 $t \geq 5$
 $r, t \geq 0$

- 4.79 Consider the problem of designing the “can” formulated in Section 2.2. Write KKT conditions and solve them. Interpret the necessary conditions at the solution point graphically.
- 4.80 A minimum weight tubular column design problem is formulated in Section 2.7 using mean radius R and thickness t as design variables. Solve the KKT conditions for the problem imposing an additional constraint $R/t \leq 50$ for the following data: $P = 50 \text{ kN}$, $l = 5.0 \text{ m}$, $E = 210 \text{ GPa}$, $\sigma_a = 250 \text{ MPa}$ and $\rho = 7850 \text{ kg/m}^3$. Interpret the necessary conditions at the solution point graphically.
- 4.81 A minimum weight tubular column design problem is formulated in Section 2.7 using outer radius R_o and inner radius R_i as design variables. Solve the KKT conditions for the problem imposing an additional constraint $0.5(R_o + R_i)/(R_o - R_i) \leq 50$. Use the same data as in Exercise 4.80. Interpret the necessary conditions at the solution point graphically.
- 4.82 An engineering design problem is formulated as
 minimize $f(x_1, x_2) = x_1^2 + 320x_1x_2$
 subject to $\frac{1}{60x_2}x_1 - 1 \leq 0$
 $1 - \frac{1}{3600}x_1(x_1 - x_2) \leq 0$
 $x_1, x_2 \geq 0$

Write KKT necessary conditions and solve for the candidate minimum designs. Verify the solutions graphically. Interpret the KKT conditions on the graph for the problem.

Formulate and solve the following problems graphically. Verify the KKT conditions at the solution point and show gradients of the cost function and active constraints on the graph.

- | | | | | | |
|------|---------------|------|---------------|------|---------------|
| 4.83 | Exercise 2.1 | 4.84 | Exercise 2.2 | 4.85 | Exercise 2.3 |
| 4.86 | Exercise 2.4 | 4.87 | Exercise 2.5 | 4.88 | Exercise 2.6 |
| 4.89 | Exercise 2.7 | 4.90 | Exercise 2.8 | 4.91 | Exercise 2.9 |
| 4.92 | Exercise 2.10 | 4.93 | Exercise 2.11 | 4.94 | Exercise 2.12 |
| 4.95 | Exercise 2.13 | 4.96 | Exercise 2.14 | | |

Section 4.5 Physical Meaning of Lagrange Multipliers

Solve the following problems graphically, verify the KKT necessary conditions for the solution points and study the effect on the cost function of changing the boundary of the active constraint(s) by one unit.

- | | | | | | |
|-------|---------------|-------|---------------|-------|---------------|
| 4.97 | Exercise 4.44 | 4.98 | Exercise 4.45 | 4.99 | Exercise 4.46 |
| 4.100 | Exercise 4.47 | 4.101 | Exercise 4.48 | 4.102 | Exercise 4.49 |
| 4.103 | Exercise 4.50 | 4.104 | Exercise 4.51 | 4.105 | Exercise 4.52 |
| 4.106 | Exercise 4.53 | 4.107 | Exercise 4.54 | 4.108 | Exercise 4.55 |
| 4.109 | Exercise 4.56 | 4.110 | Exercise 4.57 | 4.111 | Exercise 4.58 |
| 4.112 | Exercise 4.59 | 4.113 | Exercise 4.60 | 4.114 | Exercise 4.61 |

- | | | |
|---------------------|---------------------|---------------------|
| 4.115 Exercise 4.62 | 4.116 Exercise 4.63 | 4.117 Exercise 4.64 |
| 4.118 Exercise 4.65 | 4.119 Exercise 4.66 | 4.120 Exercise 4.67 |
| 4.121 Exercise 4.68 | 4.122 Exercise 4.69 | 4.123 Exercise 4.70 |
| 4.124 Exercise 4.71 | 4.125 Exercise 4.72 | 4.126 Exercise 4.73 |
| 4.127 Exercise 4.74 | 4.128 Exercise 4.75 | 4.129 Exercise 4.76 |
| 4.130 Exercise 4.77 | 4.131 Exercise 4.78 | |

Section 4.6 Global Optimality

4.132 *Answer True or False.*

1. A linear inequality constraint always defines a convex feasible region.
 2. A linear equality constraint always defines a convex feasible region.
 3. A nonlinear equality constraint cannot give a convex feasible region.
 4. A function is convex if and only if its Hessian is positive definite everywhere.
 5. An optimum design problem is convex if all constraints are linear and cost function is convex.
 6. A convex programming problem always has an optimum solution.
 7. An optimum solution for a convex programming problem is always unique.
 8. A nonconvex programming problem cannot have global optimum solution.
 9. For a convex design problem, the Hessian of the cost function must be positive semidefinite everywhere.
 10. Checking for the convexity of a function can actually identify a domain over which the function may be convex.
- 4.133 Using the definition of a line segment given in Eq. (4.65), show that the following set is convex

$$S = \{\mathbf{x} | x_1^2 + x_2^2 - 1.0 \leq 0\}$$

4.134 Find the domain for which the following functions are convex: (i) $\sin x$, (ii) $\cos x$.

Check for convexity of the following functions. If the function is not convex everywhere, then determine the domain (feasible set S) over which the function is convex.

4.135 $f(x_1, x_2) = 3x_1^2 + 2x_1x_2 + 2x_2^2 + 7$

4.136 $f(x_1, x_2) = x_1^2 + 4x_1x_2 + x_2^2 + 3$

4.137 $f(x_1, x_2) = x_1^3 + 12x_1x_2^2 + 2x_2^3 + 5x_1^2 + 3x_2$

4.138 $f(x_1, x_2) = 5x_1 - \frac{1}{16}x_1^2x_2^2 + \frac{1}{4x_1}x_2^2$

4.139 $f(x_1, x_2) = x_1^2 + x_1x_2 + x_2^2$

4.140 $U = \frac{(21.9E+07)}{V^2C} + (3.9E+06)C + (1.0E+03)V$

4.141 Consider the problem of designing the “can” formulated in Section 2.2. Check convexity of the problem. Solve the problem graphically and check the KKT conditions at the solution point.

Formulate and check convexity of the following problems; solve the problems graphically and verify the KKT conditions at the solution point.

- 4.142 Exercise 2.1 4.143 Exercise 2.3 4.144 Exercise 2.4
 4.145 Exercise 2.5 4.146 Exercise 2.9 4.147 Exercise 2.10
 4.148 Exercise 2.12 4.149 Exercise 2.14

Section 4.7 Engineering Design Examples

4.150 The problem of minimum weight design of the symmetric three-bar truss of Fig. 2-6 is formulated as follows:

minimize $f(x_1, x_2) = 2\sqrt{2}x_1 + x_2$
 subject to the constraints

$$g_1 = \frac{1}{\sqrt{2}} \left[\frac{P_u}{x_1} + \frac{P_v}{(x_1 + \sqrt{2}x_2)} \right] - 20,000 \leq 0$$

$$g_2 = \frac{\sqrt{2}P_v}{(x_1 + \sqrt{2}x_2)} - 20,000 \leq 0$$

$$g_3 = -x_1 \leq 0$$

$$g_4 = -x_2 \leq 0$$

where x_1 is the cross-sectional area of members 1 and 3 (symmetric structure) and x_2 is the cross-sectional area of member 2, $P_u = P \cos \theta$, $P_v = P \sin \theta$, with $P > 0$ and $0 \leq \theta \leq 90$. Check for convexity of the problem for $\theta = 60^\circ$.

- 4.151 For the three-bar truss problem of Exercise 4.150, consider the case of KKT conditions with g_1 as the only active constraint. Solve the conditions for optimum solution and determine the range for the load angle θ for which the solution is valid.
- 4.152 For the three-bar truss problem of Exercise 4.150, consider the case of KKT conditions with only g_1 and g_2 as active constraints. Solve the conditions for optimum solution and determine the range for the load angle θ for which the solution is valid.
- 4.153 For the three-bar truss problem of Exercise 4.150, consider the case of KKT conditions with g_2 as the only active constraint. Solve the conditions for optimum solution and determine the range for the load angle θ for which the solution is valid.
- 4.154 For the three-bar truss problem of Exercise 4.150, consider the case of KKT conditions with g_1 and g_4 as active constraints. Solve the conditions for optimum solution and determine the range for the load angle θ for which the solution is valid.

5 More on Optimum Design Concepts

Upon completion of this chapter, you will be able to:

- Write and use an alternate form of optimality conditions for constrained problems
- Determine if the candidate points are irregular
- Check the second-order optimality conditions at the candidate minimum points for general constrained problems

In this chapter, we discuss some additional topics related to the optimality condition for constrained problems. Implications of the regularity requirements in the Karush-Kuhn-Tucker (KKT) necessary conditions are discussed. Second-order optimality conditions for the problem are presented and discussed. These topics are usually not covered in a first course on optimization or in a first reading of the book. They are more suitable for a second course or a graduate level course on the subject.

5.1 Alternate Form of KKT Necessary Conditions

There is an alternate but entirely equivalent form for the KKT necessary conditions. In this form, the slack variables are not added to the inequality constraints and the conditions of Eqs. (4.46) to (4.51) are written without them. It can be seen that in the necessary conditions of Eqs. (4.46) to (4.51), the slack variable s_i appears only in two equations: Eq. (4.48) as $g_i(\mathbf{x}^*) + s_i^2 = 0$, and Eq. (4.50) as $u_i^* s_i = 0$. We shall show that both the equations can be written in an equivalent form without the slack variable s_i^2 .

Consider first Eq. (4.48), $g_i(\mathbf{x}^*) + s_i^2 = 0$ for $i = 1$ to m . The purpose of this equation is to ensure that at the candidate minimum point, all the inequalities remain satisfied. The equation can be written as $s_i^2 = -g_i(\mathbf{x}^*)$ and since $s_i^2 \geq 0$ ensures satisfaction of the constraint, we get $-g_i(\mathbf{x}^*) \geq 0$, or $g_i(\mathbf{x}^*) \leq 0$ for $i = 1$ to m . Thus, Eq. (4.48), $g_i(\mathbf{x}^*) + s_i^2 = 0$, can be simply replaced by $g_i(\mathbf{x}^*) \leq 0$.

The second equation involving the slack variable is Eq. (4.50), $u_i^* s_i = 0$, $i = 1$ to m . Multiplying the equation by s_i , we get $u_i^* s_i^2 = 0$. Now substituting $s_i^2 = -g_i(\mathbf{x}^*)$, we get $u_i^* g_i(\mathbf{x}^*) = 0$, $i = 1$ to m . This way the slack variable is eliminated from the equation and

TABLE 5-1 Alternate Form of KKT Necessary Conditions

Problem: minimize $f(\mathbf{x})$ subject to $h_i(\mathbf{x}) = 0, i = 1$ to p ; $g_j(\mathbf{x}) \leq 0, j = 1$ to m

1. Lagrangian function definition

$$L = f + \sum_{i=1}^p v_i h_i + \sum_{j=1}^m u_j g_j \quad (5.1)$$

2. Gradient conditions

$$\frac{\partial L}{\partial x_k} = 0; \quad \frac{\partial f}{\partial x_k} + \sum_{i=1}^p v_i^* \frac{\partial h_i}{\partial x_k} + \sum_{j=1}^m u_j^* \frac{\partial g_j}{\partial x_k} = 0; \quad k = 1 \text{ to } n \quad (5.2)$$

3. Feasibility check

$$g_j(\mathbf{x}^*) \leq 0; \quad j = 1 \text{ to } m \quad (5.3)$$

4. Switching conditions

$$u_j^* g_j(\mathbf{x}^*) = 0; \quad j = 1 \text{ to } m \quad (5.4)$$

5. Nonnegativity of Lagrange multipliers for inequalities

$$u_j^* \geq 0; \quad j = 1 \text{ to } m \quad (5.5)$$

6. Regularity check

Gradients of active constraints must be linearly independent. In such a case, the Lagrange multipliers for the constraints are unique.

the switching condition of Eq. (4.50) can be written as $u_i^* g_i(\mathbf{x}^*) = 0, i = 1$ to m . These conditions can be used to define various cases as $u_i^* = 0$ or $g_i = 0$ (instead of $s_i = 0$). Table 5-1 gives the KKT conditions of Theorem 4.6 in the alternate form without the slack variables, and Examples 5.1 and 5.2 provide an illustration of its use.

EXAMPLE 5.1 Use of Alternate form of KKT Conditions

Minimize $f(x, y) = (x - 10)^2 + (y - 8)^2$ subject to $g_1 = x + y - 12 \leq 0, g_2 = x - 8 < 0$.

Solution. The KKT conditions are

1. Lagrangian function definition:

$$L = (x - 10)^2 + (y - 8)^2 + u_1(x + y - 12) + u_2(x - 8)$$

2. Gradient condition: $\frac{\partial L}{\partial x} = 2(x - 10) + u_1 + u_2 = 0$

$$\frac{\partial L}{\partial y} = 2(y - 8) + u_1 = 0 \quad (a)$$

3. Feasibility check: $g_1 \leq 0, g_2 \leq 0$ (b)
4. Switching conditions: $u_1 g_1 = 0, u_2 g_2 = 0$ (c)
5. Nonnegativity of Lagrange multipliers: $u_1, u_2 \geq 0$
6. Regularity check.

The switching conditions (c) give the following four cases:

1. $u_1 = 0, u_2 = 0$ (both g_1 and g_2 inactive)
2. $u_1 = 0, g_2 = 0$ (g_1 inactive, g_2 active)
3. $g_1 = 0, u_2 = 0$ (g_1 active, g_2 inactive)
4. $g_1 = 0, g_2 = 0$ (both g_1 and g_2 active)

Case 1: $u_1 = 0, u_2 = 0$ (both g_1 and g_2 inactive).

Equations (a) give the solution as, $x = 10, y = 8$. Checking feasibility of this point gives $g_1 = 6 > 0, g_2 = 2 > 0$; thus both constraints are violated and so this case does not give any candidate minimum point.

Case 2: $u_1 = 0, g_2 = 0$ (g_1 inactive, g_2 active).

$g_2 = 0$ gives $x = 8$. Equations (a) give $y = 8$ and $u_2 = 4$. At the point $(8, 8)$, $g_1 = 4 > 0$ which is a violation. Thus the point $(8, 8)$ is infeasible and this case also does not give any candidate minimum points.

Case 3: $g_1 = 0, u_2 = 0$ (g_1 active, g_2 inactive).

Equations (a) and $g_1 = 0$ give $x = 7, y = 5, u_1 = 6 > 0$. Checking feasibility, $g_2 = -1 < 0$ which is satisfied. Since there is only one active constraint, the question of linear dependence of gradients of active constraints does not arise; therefore regularity condition is satisfied. Thus point $(7, 5)$ satisfies all the KKT necessary conditions.

Case 4: $g_1 = 0, g_2 = 0$ (both g_1 and g_2 active).

$g_1 = 0, g_2 = 0$ give $x = 8, y = 4$. Equations (a) give $u_1 = 8, u_2 = -4 < 0$, which is a violation of the necessary conditions. Therefore, this case also does not give any candidate minimum points.

It may be checked that this is a convex programming problem since constraints are linear and the cost function is convex. Therefore the point obtained in Case 3 is indeed a global minimum point according to the convexity results of Section 4.6.

EXAMPLE 5.2 Check for KKT Necessary Conditions

An optimization problem has one equality constraint h and one inequality constraint g . Check KKT necessary conditions at what is believed to be the optimum point using the following information:

$$h = 0, g = 0, \nabla f = (2, 3, 2), \nabla h = (1, -1, 1), \nabla g = (-1, -2, -1) \quad (a)$$

Solution. At the candidate minimum point, the gradients of h and g are linearly independent, so the given point is regular. The KKT conditions are

$$\begin{aligned} \nabla L &= \nabla f + v \nabla h + u \nabla g = 0 \\ h &= 0, \quad g \leq 0, \quad u g = 0, \quad u \geq 0 \end{aligned} \quad (b)$$

Substituting for ∇f , ∇h and ∇g , we get the following three equations:

$$2 + v - u = 0, \quad 3 - v - 2u = 0, \quad 2 + v - u = 0$$

These are three equations in two unknowns; however, only two of them are linearly independent. Solving for u and v , we get $u = \frac{5}{3} \geq 0$ and $v = -\frac{1}{3}$. Thus, all the KKT necessary conditions are satisfied.

5.2 Irregular Points

In all the examples that have been considered thus far it is implicitly assumed that conditions of the Karush-Kuhn-Tucker Theorem 4.6 or the Lagrange Theorem 4.5 are satisfied. In particular; we have assumed that \mathbf{x}^* is a *regular point* of the feasible design space. That is, gradients of all the active constraints at \mathbf{x}^* are linearly independent (i.e., they are not parallel to each other, nor any gradient can be expressed as a linear combination of others). It must be realized that necessary conditions are *applicable only if the assumption for regularity* of \mathbf{x}^* is satisfied. To show that the necessary conditions are not applicable if \mathbf{x}^* is not a regular point, we consider Example 5.3.

EXAMPLE 5.3 Check for KKT Conditions at Irregular Points

Minimize $f(x_1, x_2) = x_1^2 + x_2^2 - 4x_1 + 4$ subject to $g_1 = -x_1 \leq 0$, $g_2 = -x_2 \leq 0$, $g_3 = x_2 - (1 - x_1)^3 \leq 0$. Check if the minimum point $(1, 0)$ satisfies KKT necessary conditions (McCormick, 1967).

Solution. The graphical solution shown in Fig. 5-1 gives the global minimum for the problem at $\mathbf{x}^* = (1, 0)$. Let us see if the solution satisfies KKT necessary conditions:

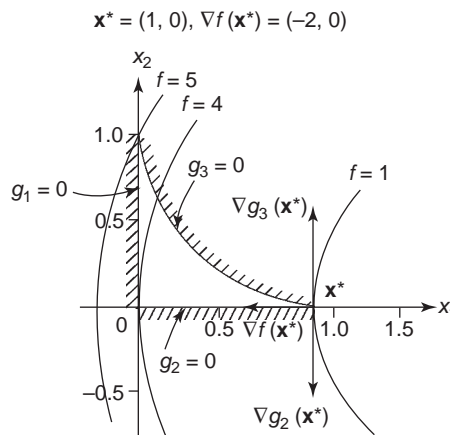


FIGURE 5-1 Graphical solution for Example 5.3. Irregular optimum point.

1. Lagrangian definition: $L = x_1^2 + x_2^2 - 4x_1 + 4 + u_1(-x_1) + u_2(-x_2) + u_3(x_2 - [1 - x_1]^3)$
2. Gradient condition: $\frac{\partial L}{\partial x_1} = 2x_1 - 4 - u_1 + u_3(3)(1 - x_1)^2 = 0$
 $\frac{\partial L}{\partial x_2} = 2x_2 - u_2 + u_3 = 0$ (a)
3. Feasibility check: $g_i \leq 0, i = 1, 2, 3$ (b)
4. Switching conditions: $u_i g_i = 0, i = 1, 2, 3$ (c)
5. Nonnegativity of Lagrange multipliers: $u_i \geq 0, i = 1, 2, 3$
6. Regularity check.

At $\mathbf{x}^* = (1, 0)$ the first constraint (g_1) is inactive and the second and third constraints are active. The switching conditions (c) identify the case as $u_1 = 0, g_2 = 0, g_3 = 0$. Substituting the solution into Eq. (a), we find that the first equation gives $2 = 0$ and therefore it is not satisfied. Thus, KKT necessary conditions are not satisfied at the minimum point.

This apparent *contradiction* can be resolved by checking the regularity condition at the minimum point $\mathbf{x}^* = (1, 0)$. The gradients of the active constraints g_2 and g_3 are given as

$$\nabla g_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}; \quad \nabla g_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

These vectors are not linearly independent. They are along the same line but in opposite directions, as shown in Fig. 5-1. Thus \mathbf{x}^* is not a regular point of the feasible set. Since this is assumed in the KKT conditions, their use is invalid here. Note also that geometrical interpretation of the KKT conditions is violated; that is, ∇f at $(1, 0)$ cannot be written as a linear combination of the gradients of the active constraints g_2 and g_3 . Actually ∇f is normal to both ∇g_2 and ∇g_3 as shown in the figure.

Note that for some problems, irregular points can be obtained as solution of the KKT conditions; however, in such cases, the Lagrange multipliers of the active constraints cannot be guaranteed to be unique. Also the constraint variation sensitivity result of Section 4.5 may or may not be applicable for some values of the Lagrange multipliers.

5.3 Second-Order Conditions for Constrained Optimization

Solutions of the necessary conditions are candidate local minimum designs. In this section, we shall discuss second-order necessary and sufficiency conditions for constrained optimization problems. As in the unconstrained case, *second-order information* about the functions at the candidate point \mathbf{x}^* will be used to determine if it is indeed a local minimum. Recall for the unconstrained problem that the local sufficiency of Theorem 4.4 requires the quadratic part of the Taylor's expansion for the function at \mathbf{x}^* to be positive for all nonzero changes \mathbf{d} . *In the constrained case, we must also consider active constraints at \mathbf{x}^* to determine feasible changes \mathbf{d} .* We will consider only the points $\mathbf{x} = \mathbf{x}^* + \mathbf{d}$ in the neighborhood of \mathbf{x}^* that satisfy the active constraint equations. *Any $\mathbf{d} \neq \mathbf{0}$ satisfying active constraints to the first order must be in the constraint tangent hyperplane* (Fig. 5-2). Such \mathbf{d} 's are then orthogonal to the gradients of the active constraints since constraint gradients are normal to the constraint tangent hyperplane. Therefore, the dot product of \mathbf{d} with each of the constraint

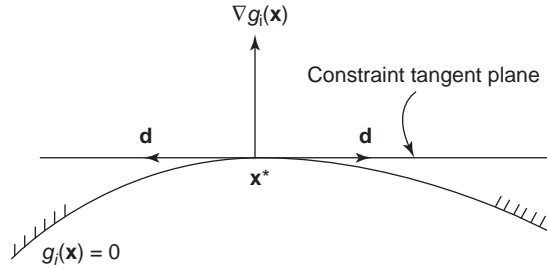


FIGURE 5-2 Directions \mathbf{d} used in constrained sufficiency conditions.

gradients ∇h_i and ∇g_i must be zero, i.e., $\nabla h_i^T \mathbf{d} = 0$ and $\nabla g_i^T \mathbf{d} = 0$. These equations are used to determine directions \mathbf{d} that define a feasible region around the point \mathbf{x}^* . Note that only active inequality constraints ($g_i = 0$) are used in determining \mathbf{d} . The situation is depicted in Fig. 5-2 for one inequality constraint.

To derive the second-order conditions, we write Taylor's expansion of the Lagrange function and consider only those \mathbf{d} that satisfy the preceding conditions. \mathbf{x}^* is then a local minimum point if the second-order term of Taylor's expansion is positive for all \mathbf{d} in the constraint tangent hyperplane. This is then the sufficient condition for an isolated local minimum point. As a necessary condition the second-order term must be nonnegative. We summarize these results in Theorems 5.1 and 5.2.

Theorem 5.1 Second-order Necessary Condition for General Constrained Problems Let \mathbf{x}^* satisfy the first-order KKT necessary conditions for the general optimum design problem. Define the Hessian of the Lagrange function L at \mathbf{x}^* as

$$\nabla^2 L = \nabla^2 f + \sum_{i=1}^p v_i^* \nabla^2 h_i + \sum_{i=1}^m u_i^* \nabla^2 g_i \quad (5.6)$$

Let there be nonzero feasible directions, $\mathbf{d} \neq \mathbf{0}$, satisfying the following linear systems at the point \mathbf{x}^* :

$$\nabla h_i^T \mathbf{d} = 0; \quad i = 1 \text{ to } p \quad (5.7)$$

$$\nabla g_i^T \mathbf{d} = 0; \text{ for all active inequalities (i.e., for those } i \text{ with } g_i(\mathbf{x}^*) = 0) \quad (5.8)$$

Then if \mathbf{x}^* is a local minimum point for the optimum design problem, it must be true that

$$Q \geq 0 \quad \text{where} \quad Q = \mathbf{d}^T \nabla^2 L(\mathbf{x}^*) \mathbf{d} \quad (5.9)$$

Note that any point that does not satisfy the second-order necessary conditions cannot be a local minimum point.

Theorem 5.2 Sufficient Conditions for General Constrained Problems Let \mathbf{x}^* satisfy the first-order KKT necessary conditions for the general optimum design problem. Define Hessian of the Lagrange function L at \mathbf{x}^* as in Eq. (5.6). Define nonzero feasible directions, $\mathbf{d} \neq \mathbf{0}$, as solutions of the linear systems

$$\nabla h_i^T \mathbf{d} = 0; \quad i = 1 \text{ to } p \quad (5.10)$$

$$\nabla g_i^T \mathbf{d} = 0 \text{ for all active inequalities with } u_i^* > 0 \quad (5.11)$$

Also let $\nabla g_i^T \mathbf{d} \leq 0$ for those active inequalities with $u_i^* = 0$. If

$$Q > 0, \quad \text{where } Q = \mathbf{d}^T \nabla^2 L(\mathbf{x}^*) \mathbf{d} \quad (5.12)$$

then \mathbf{x}^* is an *isolated local minimum* point (isolated means that there are no other local minimum points in the neighborhood of \mathbf{x}^*).

Note first the difference in the conditions for the directions \mathbf{d} in Eq. (5.8) for the necessary condition and Eq. (5.11) for the sufficient condition. In Eq. (5.8) all active inequalities with nonnegative multipliers are included whereas in Eq. (5.11) only those active inequalities with a positive multiplier are included. Equations (5.10) and (5.11) simply say that the dot product of vectors ∇h_i and \mathbf{d} and ∇g_i (having $u_i^* > 0$) and \mathbf{d} should be zero. So, only the \mathbf{d} orthogonal to the gradients of equality and active inequality constraints with $u_i^* > 0$ are considered. Or, stated differently, only \mathbf{d} in the tangent hyperplane to the active constraints at the candidate minimum point are considered. Equation (5.12) says that the Hessian of the Lagrangian must be positive definite for all \mathbf{d} lying in the constraint tangent hyperplane. Note that ∇h_i , ∇g_i and $\nabla^2 L$ are calculated at the candidate local minimum points \mathbf{x}^* satisfying the KKT necessary conditions.

It is important to note that if matrix $\nabla^2 L(\mathbf{x}^)$ is negative definite or negative semidefinite then the second-order necessary condition for a local minimum is violated and \mathbf{x}^* cannot be a local minimum point. Also if $\nabla^2 L(\mathbf{x}^*)$ is positive definite, i.e., Q in Eq. (5.12) is positive for any $\mathbf{d} \neq \mathbf{0}$ then \mathbf{x}^* satisfies the sufficiency condition for an isolated local minimum and no further checks are needed.* The reason is that if $\nabla^2 L(\mathbf{x}^*)$ is positive definite, then it is also positive definite for those \mathbf{d} that satisfy Eqs. (5.10) and (5.11). However, if $\nabla^2 L(\mathbf{x}^*)$ is not positive definite then we cannot conclude that \mathbf{x}^* is not an isolated local minimum. We must calculate \mathbf{d} to satisfy Eqs. (5.10) and (5.11) and carry out the sufficiency test given in the Theorem 5.2. This result is summarized in Theorem 5.3.

Theorem 5.3 Strong Sufficient Condition Let \mathbf{x}^* satisfy the first-order KKT necessary conditions for the general optimum design problem. Define Hessian $\nabla^2 L(\mathbf{x}^*)$ for the Lagrange function at \mathbf{x}^* as in Eq. (5.6). Then if $\nabla^2 L(\mathbf{x}^*)$ is positive definite, \mathbf{x}^* is an isolated minimum point.

It should also be emphasized that if the inequality in Eq. (5.12) is not satisfied, we cannot conclude that \mathbf{x}^* is not a local minimum. It may still be a local minimum but not an isolated one. Note also that the theorem cannot be used for any \mathbf{x}^* if its assumptions are not satisfied. In that case, we cannot draw any conclusions for the point \mathbf{x}^* .

One case arising in some applications needs special mention. This occurs when the total number of active constraints (with at least one inequality) at the candidate minimum point \mathbf{x}^ is equal to the number of independent design variables; that is, there are no design degrees of freedom.* Since \mathbf{x}^* satisfies KKT conditions, gradients of all the active constraints are linearly independent. Thus, the only solution for the system of Eqs. (5.10) and (5.11) is $\mathbf{d} = \mathbf{0}$ and Theorem 5.2 cannot be used. However, since $\mathbf{d} = \mathbf{0}$ is the only solution, there are no feasible directions in the neighborhood that can reduce the cost function any further. Thus, the point \mathbf{x}^* is indeed a local minimum for the cost function (see also the definition of a local minimum in Section 4.1.1). We consider Examples 5.4 to 5.6 to illustrate the use of sufficient conditions of optimality.

EXAMPLE 5.4 Check for Sufficient Conditions

Check sufficiency condition for Example 4.30: Minimize $f(\mathbf{x}) = \frac{1}{3}x^3 - \frac{1}{2}(b+c)x^2 + bcx + f_0$ subject to $a \leq x \leq d$ where $0 < a < b < c < d$ and f_0 are specified constants.

Solution. There is only one constrained candidate local minimum point, $x = a$. Since there is only one design variable and one active constraint, the condition $\nabla g_1 \bar{d} = 0$ of Eq. (5.11) gives $\bar{d} = 0$ as the only solution (note that \bar{d} is used as a direction for sufficiency check since d is used as a constant in the example). Therefore, Theorem 5.2 cannot be used for a sufficiency check. Also note that at $x = a$, $d^2L/dx^2 = 2a - b - c$ which can be positive, negative, or zero depending on the values of a , b , and c . So, we cannot use curvature of Hessian to check the sufficiency condition (Strong Sufficient Theorem 5.3). However, from Fig. 4-20 we observe that $x = a$ is indeed an isolated local minimum point. From this example we can conclude that if the number of active inequality constraints is equal to the number of independent design variables and all other KKT conditions are satisfied, then the candidate point is indeed a local minimum design.

EXAMPLE 5.5 Check for Sufficient Conditions

Consider the optimization problem of Example 4.31: Minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - 3x_1x_2$ subject to $g(\mathbf{x}) = x_1^2 + x_2^2 - 6 \leq 0$. Check for sufficient conditions for the candidate minimum points.

Solution. The points satisfying KKT necessary conditions are

$$(i) \mathbf{x}^* = (0, 0), u^* = 0 \quad (ii) \mathbf{x}^* = (\sqrt{3}, \sqrt{3}), u^* = \frac{1}{2} \quad (iii) \mathbf{x}^* = (-\sqrt{3}, -\sqrt{3}), u^* = \frac{1}{2} \quad (a)$$

It was previously observed in Example 4.31 and Fig. 4-21 that the point $(0, 0)$ did not satisfy the sufficiency condition, and the other two points did satisfy it. Those geometrical observations shall be mathematically verified using the sufficient theorems of optimality. The Hessian matrices for the cost and constraint functions are

$$\nabla^2 f = \begin{bmatrix} 2 & -3 \\ -3 & 2 \end{bmatrix}, \quad \nabla^2 g = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad (b)$$

By the method of Appendix B, eigenvalues of $\nabla^2 g$ are $\lambda_1 = 2$ and $\lambda_2 = 2$. Since both eigenvalues are positive, the function g is convex, and so the feasible set defined by $g(\mathbf{x}) \leq 0$ is convex by Theorem 4.9. However, since eigenvalues of $\nabla^2 f$ are -1 and 5 , f is not convex. Therefore, it cannot be classified as a convex programming problem and sufficiency cannot be shown by Theorem 4.11. We must resort to the general sufficiency Theorem 5.2. The Hessian of the Lagrangian function is given as

$$\nabla^2 L = \nabla^2 f + u \nabla^2 g = \begin{bmatrix} 2+2u & -3 \\ -3 & 2+2u \end{bmatrix} \quad (c)$$

For the first point $\mathbf{x}^* = (0, 0)$, $u^* = 0$, $\nabla^2 L$ becomes $\nabla^2 f$ (the constraint $g(\mathbf{x}) \leq 0$ is inactive). In this case the problem is unconstrained and the local sufficiency requires $\mathbf{d}^T \nabla^2 f(\mathbf{x}^*) \mathbf{d} > 0$ for all \mathbf{d} . Or, $\nabla^2 f$ should be positive definite at \mathbf{x}^* . Since both eigenvalues of $\nabla^2 f$ are not positive, we conclude that the above condition is not satisfied. Therefore, $\mathbf{x}^* = (0, 0)$ does not satisfy the second-order sufficiency condition. Note that since $\lambda_1 = -1$ and $\lambda_2 = 5$, the matrix $\nabla^2 f$ is indefinite at \mathbf{x}^* . Therefore the

point $\mathbf{x}^* = (0, 0)$ violates the second-order necessary condition of Theorem 4.4 requiring $\nabla^2 f$ to be positive semidefinite or definite at the candidate local minimum point. Thus, $\mathbf{x}^* = (0, 0)$ cannot be a local minimum point. This agrees with graphical observation made in Example 4.31.

At points $\mathbf{x}^* = (\sqrt{3}, \sqrt{3}), u^* = \frac{1}{2}$ and $\mathbf{x}^* = (-\sqrt{3}, -\sqrt{3}), u^* = \frac{1}{2}$,

$$\nabla^2 L = \nabla^2 f + u \nabla^2 g = \begin{bmatrix} 2+2u & -3 \\ -3 & 2+2u \end{bmatrix} \quad (d)$$

$$\nabla g = \pm(2\sqrt{3}, 2\sqrt{3}) = \pm 2\sqrt{3}(1, 1) \quad (e)$$

It may be checked that $\nabla^2 L$ is not positive definite at either of the two points. Therefore, we cannot use Theorem 5.3 to conclude that \mathbf{x}^* is a minimum point. We must find \mathbf{d} satisfying Eqs. (5.10) and (5.11). If we let $\mathbf{d} = (d_1, d_2)$, then $\nabla g^T \mathbf{d} = 0$ gives

$$\pm 2\sqrt{3} \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = 0; \quad \text{or } d_1 + d_2 = 0 \quad (f)$$

Thus, $d_1 = -d_2 = c$, where $c \neq 0$ is an arbitrary constant, and a $\mathbf{d} \neq \mathbf{0}$ satisfying $\nabla g^T \mathbf{d} = 0$ is given as $\mathbf{d} = c(1, -1)$. The sufficiency condition of Eq. (5.12) gives

$$Q = \mathbf{d}^T (\nabla^2 L) \mathbf{d} = c \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 3 & -3 \\ -3 & 3 \end{bmatrix} c \begin{bmatrix} 1 \\ -1 \end{bmatrix} = 12c^2 > 0 \text{ for } c \neq 0 \quad (g)$$

The points $\mathbf{x}^* = (\sqrt{3}, \sqrt{3})$ and $\mathbf{x}^* = (-\sqrt{3}, -\sqrt{3})$ satisfy the sufficiency conditions. They are therefore isolated local minimum points as was observed graphically in Example 4.31 and Fig. 4-21. We see for this example that $\nabla^2 L$ is not positive definite, but \mathbf{x}^* is still an isolated minimum point.

Note that since f is continuous and the feasible set is closed and bounded, we are guaranteed the existence of a global minimum by the Weierstrass Theorem 4.1. Also we have examined every possible point satisfying necessary conditions. Therefore, we must conclude by elimination that $\mathbf{x}^* = (\sqrt{3}, \sqrt{3})$ and $\mathbf{x}^* = (-\sqrt{3}, -\sqrt{3})$ are global minimum points. The value of the cost function for both points is $f(\mathbf{x}^*) = -3$.

EXAMPLE 5.6 Check for Sufficient Conditions

Consider Example 4.32: Minimize $f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 2$ subject to $g_1 = -2x_1 - x_2 + 4 \leq 0$, $g_2 = -x_1 - 2x_2 + 4 \leq 0$. Check the sufficiency condition for the candidate minimum point.

Solution. The KKT necessary conditions are satisfied for the point

$$x_1^* = \frac{4}{3}, \quad x_2^* = \frac{4}{3}, \quad u_1^* = \frac{2}{9}, \quad u_2^* = \frac{2}{9} \quad (a)$$

Since all the constraint functions are linear, the feasible set S is convex. The Hessian of the cost function is positive definite. Therefore, it is also convex and the problem

is convex. By Theorem 4.11, $x_1^* = \frac{4}{3}, x_2^* = \frac{4}{3}$ satisfies sufficiency conditions for a global minimum with the cost function as $f(\mathbf{x}^*) = \frac{2}{9}$.

Note that local sufficiency cannot be shown by the method of Theorem 5.2. The reason is that the conditions of Eq. (5.11) give two equations in two unknowns:

$$-2d_1 - d_2 = 0, \quad -d_1 - 2d_2 = 0 \quad (\text{b})$$

This is a homogeneous system of equations with a nonsingular coefficient matrix. Therefore, its only solution is $d_1 = d_2 = 0$. Thus, we cannot find a $\mathbf{d} \neq \mathbf{0}$ for use in the condition of Eq. (5.12), and Theorem 5.2 cannot be used. However, we have seen in the foregoing and in Fig. 4-22 that the point is actually an isolated global minimum point. Since it is a two-variable problem and two inequality constraints are active at the KKT point, the condition for local minimum is satisfied.

5.4 Sufficiency Check for Rectangular Beam Design Problem

The rectangular beam problem is formulated and graphically solved in Section 3.8. The KKT necessary conditions are written and solved in Section 4.7.2. Several points that satisfy the KKT conditions are obtained. It is seen from the graphical representation of the problem that all these points are global minima for the problem; however, none of the points is an isolated minimum. Let us show that the sufficiency condition will not be satisfied for any of these points.

Cases 3, 5, and 6 in Section 4.7.2 give solutions that satisfy the KKT conditions. Cases 5 and 6 have two active constraints; however, only the constraint with positive multiplier needs to be considered in Eq. (5.11). The sufficiency theorem requires only constraints with $u_i > 0$ to be considered in calculating the feasible directions for use in Eq. (5.12). Therefore only the g_2 constraint needs to be included in the check for sufficient conditions. Thus, *all the three cases have the same sufficiency check*. We need to calculate Hessians of the cost function and the second constraint:

$$\nabla^2 f = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \nabla^2 g_2 = \frac{(2.25E + 05)}{b^3 d^3} \begin{bmatrix} 2d^2 & bd \\ bd & 2b^2 \end{bmatrix} \quad (\text{a})$$

Since $bd = (1.125E + 05)$, $\nabla^2 g_2$ becomes

$$\nabla^2 g_2 = 2 \begin{bmatrix} \frac{2}{b^2} & (1.125E + 05)^{-1} \\ (1.125E + 05)^{-1} & \frac{2}{d^2} \end{bmatrix} \quad (\text{b})$$

The Hessian of the Lagrangian is given as

$$\nabla^2 L = \nabla^2 f + u_2 \nabla^2 g_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + 2(56250) \begin{bmatrix} \frac{2}{b^2} & (1.125E + 05)^{-1} \\ (1.125E + 05)^{-1} & \frac{2}{d^2} \end{bmatrix} \quad (\text{c})$$

$$\nabla^2 L = \begin{bmatrix} \frac{(2.25E+05)}{b^2} & 2 \\ 2 & \frac{(2.25E+05)}{d^2} \end{bmatrix} \quad (d)$$

The determinant of $\nabla^2 L$ is 0 for $bd = (1.125E + 05)$; the matrix is only positive semidefinite. Therefore, Theorem 5.3 cannot be used to show sufficiency of \mathbf{x}^* . We must check the sufficiency condition of Eq. (5.12). In order to do that, we must find directions \mathbf{y} satisfying Eq. (5.11). The gradient of g_2 is given as

$$\nabla g_2 = \left[\frac{-(2.25E+05)}{b^2 d}, \frac{-(2.25E+05)}{bd^2} \right] \quad (e)$$

The feasible directions \mathbf{y} are given by $\nabla g_2^T \mathbf{y} = 0$, as

$$\frac{1}{b} y_1 + \frac{1}{d} y_2 = 0, \quad \text{or } y_2 = -\frac{d}{b} y_1 \quad (f)$$

Therefore, vector \mathbf{y} is given as $\mathbf{y} = (1, -d/b)c$, where $c = y_1$ is any constant. Using $\nabla^2 L$ and \mathbf{y} , Q of Eq. (5.12) is given as

$$Q = \mathbf{y}^T \nabla^2 L \mathbf{y} = 0 \quad (g)$$

Thus, the sufficiency condition of Theorem 5.2 is not satisfied. The points satisfying $bd = (1.125E + 05)$ need not be isolated minimum points. This is, of course, true from Figure 3-11. Note, however, that since $Q = 0$, the second-order necessary condition of Theorem 5.1 is satisfied for Case 3 solution. Theorem 5.2 cannot be used for solutions of Cases 5 and 6 since there are two active constraints for this two variable problem; therefore there are no nonzero \mathbf{d} vectors.

It is important to note that this problem does not satisfy the condition for a convex programming problem and all the points satisfying KKT conditions do not satisfy the sufficiency condition for isolated minimum. Yet, all the points are actually global minimum points. Two conclusions can be drawn from this example:

1. *Global optimum solutions* can be obtained for problems that cannot be classified as convex programming problems. We cannot show global optimality unless we find all the local optimum solutions in the closed and bounded set (Weierstrass Theorem 4.1).
2. *If sufficiency conditions are not satisfied*, the only conclusion we can draw is that the candidate point need not be an isolated minimum. It may have many local optima in the neighborhood, and they may all be actually global solutions.

Exercises for Chapter 5

5.1 *Answer True or False.*

1. A convex programming problem always has a unique global minimum point.
2. For a convex programming problem, KKT necessary conditions are also sufficient.
3. The Hessian of the Lagrange function must be positive definite at constrained minimum points.

4. For a constrained problem, if the sufficiency condition of Theorem 5.2 is violated, the candidate point \mathbf{x}^* may still be a minimum point.
 5. If the Hessian of the Lagrange function at \mathbf{x}^* , $\nabla^2 L(\mathbf{x}^*)$, is positive definite, the optimum design problem is convex.
 6. For a constrained problem, the sufficient condition at \mathbf{x}^* is satisfied if there are no feasible directions in a neighborhood of \mathbf{x}^* along which the cost function reduces.
- 5.2 Formulate the problem of Exercise 4.84. Show that the solution point for the problem is not a regular point. Write KKT conditions for the problem, and study the implication of the irregularity of the solution point.

- 5.3 Solve the following problem using the graphical method:

$$\begin{aligned} \text{Minimize } f(x_1, x_2) &= (x_1 - 10)^2 + (x_2 - 5)^2 \\ \text{subject to } x_1 + x_2 &\leq 12, x_1 \leq 8, x_1 - x_2 \leq 4 \end{aligned}$$

Show that the minimum point does not satisfy the regularity condition. Study the implications of this situation.

Solve the following problems graphically. Check necessary and sufficient conditions for candidate local minimum points and verify them on the graph for the problem.

- 5.4 Minimize $f(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8x_1$
subject to $x_1 + x_2 = 4$

- 5.5 Maximize $F(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8x_1$
subject to $x_1 + x_2 = 4$

- 5.6 Minimize $f(x_1, x_2) = (x_1 - 2)^2 + (x_2 + 1)^2$
subject to $2x_1 + 3x_2 - 4 = 0$

- 5.7 Minimize $f(x_1, x_2) = 4x_1^2 + 9x_2^2 + 6x_2 - 4x_1 + 13$
subject to $x_1 - 3x_2 + 3 = 0$

- 5.8 Minimize $f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 + 2)^2 + (x_3 - 2)^2$
subject to $2x_1 + 3x_2 - 1 = 0$
 $x_1 + x_2 + 2x_3 - 4 = 0$

- 5.9 Minimize $f(x_1, x_2) = 9x_1^2 + 18x_1x_2 + 13x_2^2 - 4$
subject to $x_1^2 + x_2^2 + 2x_1 = 16$

- 5.10 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
subject to $x_1 + x_2 - 4 = 0$

- 5.11 Minimize $f(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8$
subject to $x_1 + x_2 = 4$

- 5.12 Maximize $F(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8$
subject to $x_1 + x_2 = 4$

- 5.13 Maximize $F(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8$
subject to $x_1 + x_2 \leq 4$

- 5.14 Minimize $f(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8$
subject to $x_1 + x_2 \leq 4$
- 5.15 Maximize $F(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8x_1$
subject to $x_1 + x_2 \leq 4$
- 5.16 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
subject to $x_1 + x_2 \geq 4$
 $x_1 - x_2 - 2 = 0$
- 5.17 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
subject to $x_1 + x_2 = 4$
 $x_1 - x_2 - 2 \geq 0$
- 5.18 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
subject to $x_1 + x_2 \geq 4$
 $x_1 - x_2 \geq 2$
- 5.19 Minimize $f(x, y) = (x - 4)^2 + (y - 6)^2$
subject to $12 \geq x + y$
 $x \geq 6, y \geq 0$
- 5.20 Minimize $f(x_1, x_2) = 2x_1 + 3x_2 - x_1^3 - 2x_2^2$
subject to $x_1 + 3x_2 \leq 6$
 $5x_1 + 2x_2 \leq 10$
 $x_1, x_2 \geq 0$
- 5.21 Minimize $f(x_1, x_2) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8x_1$
subject to $x_1 + x_2 \leq 4$
- 5.22 Minimize $f(x_1, x_2) = x_1^2 + x_2^2 - 4x_1 - 2x_2 + 6$
subject to $x_1 + x_2 \geq 4$
- 5.23 Minimize $f(x_1, x_2) = 2x_1^2 - 6x_1x_2 + 9x_2^2 - 18x_1 + 9x_2$
subject to $x_1 + 2x_2 \leq 10$
 $4x_1 - 3x_2 \leq 20; x_i \geq 0; i = 1, 2$
- 5.24 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
subject to $x_1 + x_2 - 4 \leq 0$
- 5.25 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
subject to $x_1 + x_2 - 4 \leq 0$
 $x_1 - x_2 - 2 \leq 0$
- 5.26 Minimize $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 1)^2$
subject to $x_1 + x_2 - 4 \leq 0$
 $2 - x_1 \leq 0$
- 5.27 Minimize $f(x_1, x_2) = 9x_1^2 - 18x_1x_2 + 13x_2^2 - 4$
subject to $x_1^2 + x_2^2 + 2x_1 \geq 16$

- 5.28 Minimize $f(x_1, x_2) = (x_1 - 3)^2 + (x_2 - 3)^2$
 subject to $x_1 + x_2 \leq 4$
 $x_1 - 3x_2 = 1$
- 5.29 Minimize $f(x_1, x_2) = x_1^3 - 16x_1 + 2x_2 - 3x_2^2$
 subject to $x_1 + x_2 \leq 3$
- 5.30 Minimize $f(x_1, x_2) = 3x_1^2 - 2x_1x_2 + 5x_2^2 + 8x_2$
 subject to $x_1^2 - x_2^2 + 8x_2 \leq 16$
- 5.31 Minimize $f(x, y) = (x - 4)^2 + (y - 6)^2$
 subject to $x + y \leq 12$
 $x \leq 6$
 $x, y \geq 0$
- 5.32 Minimize $f(x, y) = (x - 8)^2 + (y - 8)^2$
 subject to $x + y \leq 12$
 $x \leq 6$
 $x, y \geq 0$
- 5.33 Maximize $F(x, y) = (x - 4)^2 + (y - 6)^2$
 subject to $x + y \leq 12$
 $6 \geq x$
 $x, y \geq 0$
- 5.34 Maximize $F(r, t) = (r - 8)^2 + (t - 8)^2$
 subject to $10 \geq r + t$
 $t \leq 5$
 $r, t \geq 0$
- 5.35 Maximize $F(r, t) = (r - 3)^2 + (t - 2)^2$
 subject to $10 \geq r + t$
 $t \leq 5$
 $r, t \geq 0$
- 5.36 Maximize $F(r, t) = (r - 8)^2 + (t - 8)^2$
 subject to $r + t \leq 10$
 $t \geq 0$
 $r \geq 0$
- 5.37 Maximize $F(r, t) = (r - 3)^2 + (t - 2)^2$
 subject to $10 \geq r + t$
 $t \geq 5$
 $r, t \geq 0$
- 5.38 Formulate and graphically solve Exercise 2.23 of the design of a cantilever beam using hollow circular cross section. Check the necessary and sufficient conditions at the optimum point. The data for the problem are $P = 10 \text{ kN}$; $l = 5 \text{ m}$; modulus of elasticity, $E = 210 \text{ GPa}$; allowable bending stress, $\sigma_a = 250 \text{ MPa}$; allowable shear stress, $\tau_a = 90 \text{ MPa}$; and mass density, $\rho = 7850 \text{ kg/m}^3$; $0 \leq R_o \leq 20 \text{ cm}$, and $0 \leq R_i \leq 20 \text{ cm}$.

- 5.39 Formulate and graphically solve Exercise 2.24. Check the necessary and sufficient conditions for the solution points and verify them on the graph.
- 5.40 Formulate and graphically solve Exercise 3.28. Check the necessary and sufficient conditions for the solution points and verify them on the graph.

Find optimum solutions for the following problems graphically. Check necessary and sufficient conditions for the solution points and verify them on the graph for the problem.

- 5.41 A minimum weight tubular column design problem is formulated in Section 2.7 using mean radius R and thickness t as design variables. Solve the problem by imposing an additional constraint $R/t \leq 50$ for the following data: $P = 50 \text{ kN}$, $l = 5.0 \text{ m}$, $E = 210 \text{ GPa}$, $\sigma_a = 250 \text{ MPa}$, and $\rho = 7850 \text{ kg/m}^3$.
- 5.42 A minimum weight tubular column design problem is formulated in Section 2.7 using outer radius R_o and inner radius R_i as design variables. Solve the problem by imposing an additional constraint $0.5(R_o + R_i)/(R_o - R_i) \leq 50$. Use the same data as in Exercise 5.41.
- 5.43 Solve the problem of designing a “can” formulated in Section 2.2.
- 5.44 Exercise 2.1
- 5.45* Exercise 3.34
- 5.46* Exercise 3.35
- 5.47* Exercise 3.36
- 5.48* Exercise 3.54
- 5.49 *Answer True or False.*
1. Candidate minimum points for a constrained problem that do not satisfy second-order sufficiency conditions can be global minimum designs.
 2. Lagrange multipliers may be used to calculate the sensitivity coefficient for the cost function with respect to the right side parameters even if Theorem 4.7 cannot be used.
 3. Relative magnitudes of the Lagrange multipliers provide useful information for practical design problems.
- 5.50 A circular tank that is closed at both ends is to be fabricated to have a volume of $250\pi \text{ m}^3$. The fabrication cost is found to be proportional to the surface area of the sheet metal needed for fabrication of the tank and is $\$400/\text{m}^2$. The tank is to be housed in a shed with a sloping roof which limits the height of the tank by the relation $H \leq 8D$, where H is the height and D is the diameter of the tank. The problem is formulated as minimize $f(D, H) = 400(0.5\pi D^2 + \pi DH)$ subject to the constraints $\frac{\pi}{4} D^2 H = 250\pi$, and $H \leq 8D$. Ignore any other constraints.
1. Check for convexity of the problem.
 2. Write KKT necessary conditions.
 3. Solve KKT necessary conditions for local minimum points. Check sufficient conditions and verify the conditions graphically.
 4. What will be the change in cost if the volume requirement is changed to $255\pi \text{ m}^3$ in place of $250\pi \text{ m}^3$?
- 5.51 A symmetric (area of member 1 is the same as area of member 3) three-bar truss problem is described in Section 2.10.

1. Formulate the minimum mass design problem treating A_1 and A_2 as design variables.
2. Check for convexity of the problem.
3. Write KKT necessary conditions for the problem.
4. Solve the optimum design problem using the data: $P = 50\text{ kN}$, $\theta = 30^\circ$, $\rho = 7800\text{ kg/m}^3$, $\sigma_a = 150\text{ MPa}$. Verify the solution graphically and interpret the necessary conditions on the graph for the problem.
5. What will be the effect on the cost function if σ_a is increased to 152 MPa ?

Formulate and solve the following problems graphically; check necessary and sufficient conditions at the solution points, verify the conditions on the graph for the problem and study the effect of variations in constraint limits on the cost function.

5.52	Exercise 2.1	5.53	Exercise 2.3	5.54	Exercise 2.4
5.55	Exercise 2.5	5.56	Exercise 2.9	5.57	Exercise 4.92
5.58	Exercise 2.12	5.59	Exercise 2.14	5.60	Exercise 2.23
5.61	Exercise 2.24	5.62	Exercise 5.41	5.63	Exercise 5.42
5.64	Exercise 5.43	5.65	Exercise 3.28	5.66*	Exercise 3.34
5.67*	Exercise 3.35	5.68*	Exercise 3.36	5.69*	Exercise 3.39
5.70*	Exercise 3.40	5.71*	Exercise 3.41	5.72*	Exercise 3.46
5.73*	Exercise 3.47	5.74*	Exercise 3.48	5.75*	Exercise 3.49
5.76*	Exercise 3.50	5.77*	Exercise 3.51	5.78*	Exercise 3.52
5.79*	Exercise 3.53	5.80*	Exercise 3.54		

6 Linear Programming Methods for Optimum Design

Upon completion of this chapter, you will be able to:

- Transform a linear programming problem to the standard form
- Explain terminology and concepts related to linear programming problems
- Use the two-phase Simplex method to solve linear programming problems
- Perform postoptimality analysis for linear programming problems

An optimum design problem having linear cost and constraint functions in the design variables is called a *linear programming problem*. We shall use the abbreviation LP for linear programming problems, or simply for linear programs. LP problems arise in many fields of engineering such as water resources, systems engineering, traffic flow control, resources management, transportation engineering, and electrical engineering. In the areas of aerospace, automotive, structural, or mechanical system design, most problems are not linear. However, one way of solving nonlinear programming problems is to transform them to a sequence of linear programs (Chapter 10). Many other nonlinear programming methods also solve a linear programming problem during the iterative process. Thus, linear programming methods are useful in many applications and must be clearly understood. This chapter describes the basic theory and concepts for solving such problems.

In Section 2.11, a general mathematical model for optimum design nonlinear problems was defined to minimize a cost function subject to equality and “ \leq ” inequality constraints. In Chapter 4, a general theory of optimum design for treating the model was described. That theory can also be used to solve LP problems. However, more efficient and elegant numerical methods are available to solve the LP problem directly. Since there are numerous LP problems in the real world, it is worthwhile to discuss these methods in detail. Any linear function $f(\mathbf{x})$ of n variables is written as:

$$f(\mathbf{x}) = c_1x_1 + c_2x_2 + \dots + c_nx_n = \sum_{i=1}^n c_i x_i = \mathbf{c}^T \mathbf{x}$$

where c_i , $i = 1$ to n are constants. All functions of an LP problem can be represented in the preceding form. Therefore, the general nonlinear optimization model is replaced by a linear

model in this chapter and a standard form for the model is defined. A two-phase algorithm to solve LP problems, known as the Simplex method, is developed and illustrated with simple numerical examples.

Details of the Simplex method are described to show the numerical steps needed to solve LP problems. Before attempting to implement the algorithm into a computer program, the existence of standard packages for solving LP problems must be checked. Most information technology centers have at least one software package to treat such problems, e.g., LINDO (Schrage, 1981). It is more economical to use the available software than to develop a new one.

It is noted here that the subject of linear programming is well developed and several excellent full-length textbooks are available on the subject. These books may be consulted for in-depth treatment of the subject.

6.1 Definition of a Standard Linear Programming Problem

Linear programming problems may have equality as well as inequality constraints. Also, many problems require maximization of a function whereas others require minimization. Although the *standard LP problem* can be defined in several different ways, we define it as *minimization of a function with equality constraints and nonnegativity of design variables*. This form will be used to describe the Simplex method to solve linear programming problems. The form is not as restrictive as it may appear since all other LP problems can be readily transcribed into it. We shall explain the process of transcribing a given LP problem into the standard form.

6.1.1 Linear Constraints

The i th linear constraint involving k design variables, $y_j, j = 1$ to k has one of three possible forms, “ \leq ,” “ $=$,” or “ \geq ”:

$$\begin{aligned} a_{i1}y_1 + \dots + a_{ik}y_k &\leq b_i \\ a_{i1}y_1 + \dots + a_{ik}y_k &= b_i \\ a_{i1}y_1 + \dots + a_{ik}y_k &\geq b_i \end{aligned} \tag{6.1}$$

where a_{ij} and b_i are known constants. Also b_i , called the *resource limits*, are assumed to be always nonnegative, i.e., $b_i \geq 0$. b_i 's can always be made nonnegative by multiplying both sides of Eq. (6.1) by -1 if necessary. Note, however, that multiplication by -1 changes the sense of the original inequality, i.e., “ \leq type” becomes “ \geq type” and vice versa. For example, a constraint $y_1 + 2y_2 \leq -2$ must be transformed as $-y_1 - 2y_2 \geq 2$ to have a nonnegative right side.

Since only equality constraints are treated in the standard LP, the inequalities in Eq. (6.1) must be converted to equalities. This is no real restriction since any inequality can be converted to an equality by introducing a nonnegative *slack* or *surplus variable* as explained in the following paragraphs. Note also that since b_i 's are required to be nonnegative in Eq. (6.1), it is not always possible to convert “ \geq ” inequalities to the “ \leq form” and keep $b_i \geq 0$. In Chapters 2–5, this was done where a standard optimization problem was defined with only “ \leq type” constraints. However, in this chapter, we will have to explicitly treat “ \geq type” linear inequalities. It will be seen later that “ \geq type” constraints do require a special treatment in LP methods.

For the i th “ \leq type” constraint, we introduce a nonnegative *slack variable* $s_i \geq 0$ and convert it to an equality as

$$a_{i1}y_1 + a_{i2}y_2 + \dots + a_{ik}y_k + s_i = b_i \quad (6.2)$$

We also introduced the idea of slack variables in Chapter 4. There s_i^2 was used as a slack variable instead of s_i . That was done to avoid the additional constraint $s_i \geq 0$. However, in LP problems we cannot use s_i^2 as a slack variable because it makes the problem nonlinear. Therefore, we will use s_i as a slack variable along with the additional constraint $s_i \geq 0$. For example, a constraint $2y_1 - y_2 \leq 4$ will be transformed as $2y_1 - y_2 + s_1 = 4$ with $s_1 \geq 0$ as its slack variable.

Similarly, the i th “ \geq type” constraint is converted to equality by subtracting a nonnegative *surplus variable* $s_i \geq 0$, as

$$a_{i1}y_1 + a_{i2}y_2 + \dots + a_{ik}y_k - s_i = b_i \quad (6.3)$$

The idea of a surplus variable is very similar to the slack variable. For the “ \geq type” constraint, the left side has to be always greater than or equal to the right side, so we must subtract a nonnegative variable to transform it to an equality. For example, a constraint $-y_1 + 2y_2 \geq 2$ will be transformed as $-y_1 + 2y_2 - s_1 = 2$ with $s_1 \geq 0$ as its surplus variable. Note that slack and surplus variables are *additional unknowns* that must be determined as a part of the solution.

6.1.2 Unrestricted Variables

In addition to the equality constraints, we require all design variables to be nonnegative in the standard LP problem, i.e., $y_i \geq 0$, $i = 1$ to k . If a design variable y_j is unrestricted in sign, it can always be written as the difference of two nonnegative variables, as $y_j = y_j^+ - y_j^-$, with $y_j^+ \geq 0$ and $y_j^- \geq 0$. This decomposition is substituted into all equations and y_j^+ and y_j^- are treated as unknowns in the problem. At the optimum, if $y_j^+ \geq y_j^-$ then y_j is nonnegative, and if $y_j^+ \leq y_j^-$ then y_j is nonpositive. This treatment for each free variable increases the dimension of the design variable vector by 1.

6.1.3 Standard LP Definition

For notational clarity, let \mathbf{x} represent an n -vector consisting of k original design variables and $(n - k)$ slack, surplus, or other variables. Now let us define the standard LP problem as: find an n -vector \mathbf{x} to minimize a linear cost function

$$f = c_1x_1 + c_2x_2 + \dots + c_nx_n \quad (6.4)$$

subject to the equality constraints

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ \cdot & \quad \cdot \quad \dots \quad \cdot \quad \cdot \\ \cdot & \quad \cdot \quad \dots \quad \cdot \quad \cdot \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &= b_m \end{aligned} \quad (6.5)$$

with $b_i \geq 0$, $i = 1$ to m ; and nonnegativity constraints on the design variables

$$x_j \geq 0; \quad j = 1 \text{ to } n \quad (6.6)$$

The quantities $b_i \geq 0$, c_j , and a_{ij} ($i = 1$ to m and $j = 1$ to n) are known constants, and m and n are positive integers. Note that b_i are required to be positive or at the most zero.

The standard LP problem can also be written in the summation notation as

$$\text{minimize } f = \sum_{i=1}^n c_i x_i \quad (6.7)$$

subject to the nonnegativity constraints of Eq. (6.6) and the constraints

$$\sum_{j=1}^n a_{ij} x_j = b_i; \quad b_i \geq 0, \quad i = 1 \text{ to } m \quad (6.8)$$

Matrix notation may also be used to define the LP problem as

$$\text{minimize } f = \mathbf{c}^T \mathbf{x} \quad (6.9)$$

subject to the constraints

$$\mathbf{Ax} = \mathbf{b}; \quad \mathbf{b} \geq \mathbf{0} \quad (6.10)$$

$$\mathbf{x} \geq \mathbf{0} \quad (6.11)$$

where $\mathbf{A} = [a_{ij}]$ is an $m \times n$ matrix, \mathbf{c} and \mathbf{x} are n -vectors, and \mathbf{b} is an m -vector. Note that the vector inequalities, such as $\mathbf{b} \geq \mathbf{0}$ in Eq. (6.10), are assumed to be applied to each component of the vector throughout the text.

The formulations given in Eqs. (6.4) to (6.11) are more general than what may appear at first sight because all LP problems can be transcribed into them. Conversion of “ \leq type” and “ \geq type” inequalities to equalities using slack and surplus variables has been explained previously. Unrestricted variables can be decomposed into the difference of two nonnegative variables. *Maximization of functions* can also be routinely treated. For example, if the objective is to maximize a function (rather than minimize it), we simply minimize its negative. Maximization of a function $z = (d_1 x_1 + d_2 x_2 + \dots + d_n x_n)$ is equivalent to minimization of its negative, $f = -(d_1 x_1 + d_2 x_2 + \dots + d_n x_n)$. *Note that a function that is to be maximized is denoted as z in this chapter.* It is henceforth assumed that the LP problem has been converted into the standard form defined in Eqs. (6.4) to (6.11). Example 6.1 shows conversion to standard LP form.

EXAMPLE 6.1 Conversion to Standard LP Form

Convert the following problem into the standard LP form: maximize $z = 2y_1 + 5y_2$ subject to $3y_1 + 2y_2 \leq 12$, $2y_1 + 3y_2 \geq 6$, and $y_1 \geq 0$. y_2 is unrestricted in sign.

Solution. To transform the problem into the standard LP form, we take the following steps:

1. Since y_2 is unrestricted in sign, we split it into its positive and negative parts as $y_2 = y_2^+ - y_2^-$ with $y_2^+ \geq 0$, $y_2^- \geq 0$.
2. Substituting this definition of y_2 into the problem, we get: maximize $z = 2y_1 + 5(y_2^+ - y_2^-)$ subject to $3y_1 + 2(y_2^+ - y_2^-) \leq 12$, $2y_1 + 3(y_2^+ - y_2^-) \geq 6$, and $y_1, y_2^+, y_2^- \geq 0$.
3. The right sides of both the constraints are nonnegative so they conform to the standard form, and there is no need to modify them further.
4. Converting to a minimization problem subject to equality constraints, we get the problem in the standard form as minimize $f = -2y_1 - 5(y_2^+ - y_2^-)$ subject to $3y_1 + 2(y_2^+ - y_2^-) + s_1 = 12$, $2y_1 + 3(y_2^+ - y_2^-) - s_2 = 6$, $y_1, y_2^+, y_2^-, s_1, s_2 \geq 0$, where

s_1 = slack variable for the first constraint and s_2 = surplus variable for the second constraint.

5. We can redefine the solution variables as $x_1 = y_1$, $x_2 = y_2^+$, $x_3 = y_2^-$, $x_4 = s_1$, $x_5 = s_2$ and rewrite the problem in the standard form as

$$\text{minimize } f = -2x_1 - 5x_2 + 5x_3 \quad (\text{a})$$

subject to

$$3x_1 + 2x_2 - 2x_3 + x_4 = 12 \quad (\text{b})$$

$$2x_1 + 3x_2 - 3x_3 - x_5 = 6 \quad (\text{c})$$

$$x_i \geq 0, \quad i = 1 \text{ to } 5 \quad (\text{d})$$

Comparing the preceding equations with Eqs. (6.9) to (6.11), we can define the following quantities:

$$m = 2 \text{ (the number of equations)} \quad n = 5 \text{ (the number of variables)}$$

$$\mathbf{x} = [x_1 \ x_2 \ x_3 \ x_4 \ x_5]^T \quad \mathbf{c} = [-2 \ -5 \ 5 \ 0 \ 0]^T \quad (\text{e})$$

$$\mathbf{b} = [12 \ 6]^T \quad \mathbf{A} = [a_{ij}]_{2 \times 5} = \begin{bmatrix} 3 & 2 & -2 & 1 & 0 \\ 2 & 3 & -3 & 0 & -1 \end{bmatrix} \quad (\text{f})$$

6.2 Basic Concepts Related to Linear Programming Problems

Several terms related to LP problems are defined and explained. Some fundamental properties of LP problems are discussed. It is shown that the optimum solution for an LP problem always lies on the boundary of the feasible set. In addition, the solution is at least at one of the vertices of the convex feasible set (called the convex polyhedral set). Some LP theorems are stated and their significance is discussed. The geometrical meaning of the optimum solution is explained.

6.2.1 Basic Concepts

Since all functions are linear in an LP problem, the feasible set defined by linear equalities or inequalities is *convex* (Section 4.6). Also, the cost function is linear, so it is convex. Therefore, the LP problem is convex, and if an *optimum solution* exists, it is *global* as shown in Theorem 4.10.

Note also that even when there are inequality constraints in an LP design problem, the *solution, if it exists, always lies on the boundary of the feasible set*; i.e., some constraints are always active at the optimum. This can be seen by writing the necessary conditions of Theorem 4.4 for an unconstrained optimum. These conditions, $\partial f / \partial x_i = 0$, when used for the cost function of Eq. (6.7), give $c_i = 0$ for $i = 1$ to n . This is not possible, as all c_i 's are not zero. If all c_i 's were zero, there would be no cost function. Therefore, by contradiction, the *optimum solution for any LP problem must lie on the boundary of the feasible set*. This is in contrast to the general nonlinear problems where the optimum can be inside or on the boundary of the feasible set.

An optimum solution of the LP problem must also satisfy the equality constraints in Eq. (6.5). Only then can the solution be feasible. Therefore, to have a meaningful optimum design

problem, Eq. (6.5) should have more than one solution. Only then there is a choice of feasible solutions that can have minimum cost. To have many solutions, the number of linearly independent equations in Eq. (6.5) must be less than n , the number of variables in the LP problem (refer to Section B.5 in Appendix B for further discussion on a general solution of m equations in n unknowns). It is assumed in the following discussion that all the m rows of the matrix \mathbf{A} in Eq. (6.10) are linearly independent and that $m < n$. This means that there are no redundant equations. Therefore, Eq. (6.5) has *infinite solutions and we seek a feasible solution that also minimizes the cost function*. A method for solving simultaneous equations (6.5) based on *Gaussian elimination* is described in Appendix B. The Simplex method of LP described later in the chapter uses steps of the Gaussian elimination procedure. Therefore, that procedure must be reviewed thoroughly before studying the Simplex method.

We shall use Example 6.2 to illustrate the preceding ideas. It shall also be used later to introduce LP terminology and the basic steps of the Simplex method.

EXAMPLE 6.2 Profit Maximization Problem— Characterization of Solution for LP Problems

As an example of solving constraint equations, we consider the profit maximization problem solved graphically in Chapter 3. The problem is to find x_1 and x_2 to

$$\text{minimize } f = -400x_1 - 600x_2 \quad (\text{a})$$

subject to

$$x_1 + x_2 \leq 16 \quad (\text{b})$$

$$\frac{1}{28}x_1 + \frac{1}{14}x_2 \leq 1 \quad (\text{c})$$

$$\frac{1}{14}x_1 + \frac{1}{24}x_2 \leq 1 \quad (\text{d})$$

$$x_1, x_2 \geq 0 \quad (\text{e})$$

Solution. The graphical solution for the problem is given in Fig. 6-1. All constraints of Eqs. (b) to (e) are plotted and some isocost lines are shown. Each point of the region bounded by the polygon ABCDE satisfies all the constraints of Eqs. (b) to (d) and the nonnegativity conditions of Eq. (e). It is seen from Fig. 6-1 that the vertex D gives the optimum solution.

Introducing slack variables for constraints of Eqs. (b) to (d) and writing the problem in the standard LP form, we have

$$\text{minimize } f = -400x_1 - 600x_2 \quad (\text{f})$$

subject to

$$x_1 + x_2 + x_3 = 16 \quad (\text{g})$$

$$\frac{1}{28}x_1 + \frac{1}{14}x_2 + x_4 = 1 \quad (\text{h})$$

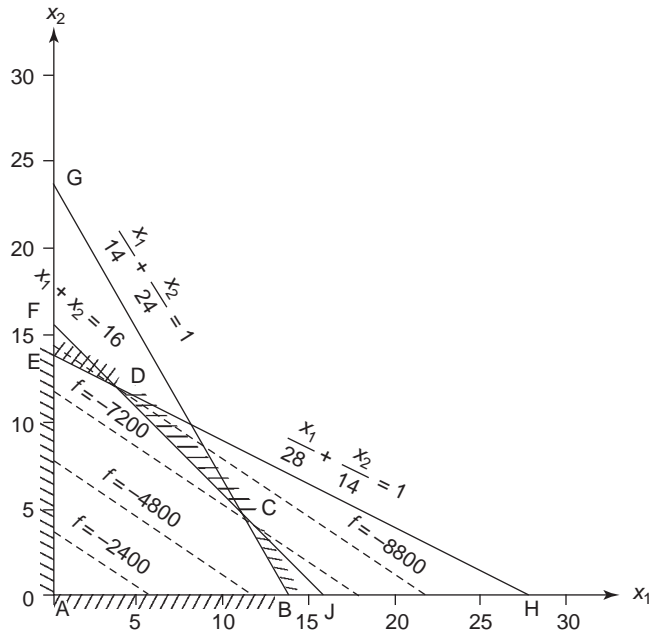


FIGURE 6-1 Graphical solution for profit maximization LP problem. Optimum point = (4, 12). Optimum cost = -8800.

$$\frac{1}{14}x_1 + \frac{1}{24}x_2 + x_5 = 1 \quad (\text{i})$$

$$x_i \geq 0, \quad i = 1 \text{ to } 5 \quad (\text{j})$$

where x_3 , x_4 , and x_5 are slack variables for the first, second, and third constraints, respectively.

Note that all three equations in Eq. (g) to (i) are linearly independent. Since the number of variables (5) exceeds the number of constraint equations (3), a unique solution cannot exist for Eqs. (g) to (i) (see Appendix B). Actually there are infinite solutions. To see this, we write a general solution for the equations by transferring the terms associated with the variables x_1 and x_2 to the right side of Eqs. (g) to (i) as

$$x_3 = 16 - x_1 - x_2 \quad (\text{k})$$

$$x_4 = 1 - \frac{1}{28}x_1 - \frac{1}{14}x_2 \quad (\text{l})$$

$$x_5 = 1 - \frac{1}{14}x_1 - \frac{1}{24}x_2 \quad (\text{m})$$

In these equations, x_1 and x_2 act as independent variables that can be given any value, and x_3 , x_4 , and x_5 are dependent on them. Different values for x_1 and x_2 generate different values for x_3 , x_4 , and x_5 . A solution of particular interest in LP problems is obtained by setting p of the variables to zero and solving for the rest, where p is the difference between the number of variables (n) and the number of constraint equations (m), i.e., $p = n - m$ [e.g., $p = 2$ in the case of Eqs. (g) to (i)]. With two variables set to zero, a unique solution of Eqs. (g) to (i) exists for the remaining three variables

since there are now three equations in three unknowns. A solution obtained by setting p variables to zero is called the basic solution. For example, a basic solution is obtained from Eqs. (g) to (i) or (k) to (m) by setting $x_1 = 0$ and $x_2 = 0$, as $x_3 = 16$, $x_4 = 1$, $x_5 = 1$. Another basic solution is obtained by setting $x_1 = 0$, $x_3 = 0$, and solving the three equations in the remaining three unknowns as $x_2 = 16$, $x_4 = -\frac{2}{7}$, $x_5 = \frac{1}{3}$. Since there are 10 different ways in which combinations of 2 variables can be set to zero, there are 10 basic solutions (later a formula is given to calculate the number of basic solutions). Table 6-1 shows all 10 basic solutions for the present example obtained by using the procedure just described. Note that of the 10 solutions, exactly 5 (Nos. 1, 2, 6, 8, and 9) correspond to the vertices of the polygon of Fig. 6-1, and the remaining 5 violate the nonnegativity condition and correspond to infeasible vertices. Therefore, only 5 of the 10 basic solutions are feasible. By moving the isocost line parallel to itself, it is seen that the optimum solution is at point D. Note that the optimum point is at one of the vertices of the feasible polygon. This will be observed later as a general property of any LP problem. That is, *if an LP has a solution, it is at least at one of the vertices of the feasible set.*

TABLE 6-1 Ten Basic Solutions for the Profit Maximization Problem

No.	x_1	x_2	x_3	x_4	x_5	f	Location in Fig. 6-1
1	0	0	16	1	1	0	A
2	0	14	2	0	$\frac{5}{12}$	-8400	E
3	0	16	0	$-\frac{2}{7}$	$\frac{1}{3}$	—	F (infeasible)
4	0	24	-8	$-\frac{5}{7}$	0	—	G (infeasible)
5	16	0	0	$\frac{3}{7}$	$-\frac{2}{7}$	—	J (infeasible)
6	14	0	2	$\frac{1}{2}$	0	-5600	B
7	28	0	-12	0	-1	—	H (infeasible)
8	4	12	0	0	$\frac{3}{14}$	-8800	D
9	11.2	4.8	0	$\frac{1}{5}$	0	-7360	C
10	$\frac{130}{17}$	$\frac{168}{17}$	$-\frac{26}{17}$	0	0	—	I (infeasible)

6.2.2 LP Terminology

We shall now introduce various definitions and terms related to the LP problem. Example 6.2 and Fig. 6-1 will be used to illustrate the meaning of these terms. Also, the definitions of convex sets, convex function, and the line segment introduced earlier in Section 4.6 will be used here.

Vertex (Extreme) Point. This is a point of the feasible set that does not lie on a line segment joining two other points of the set. For example, every point on the circumference of a circle and each vertex of the polygon satisfy the requirements for an extreme point.

Feasible Solution. Any solution of the constraint Eq. (6.5) satisfying the nonnegativity conditions is a feasible solution. In the profit maximization example of Fig. 6-1, every point bounded by the polygon ABCDE (convex set) is a feasible solution.

Basic Solution. A basic solution is a solution of the constraint Eq. (6.5) obtained by setting the “redundant number” $(n - m)$ of the variables to zero and solving the equations simultaneously for the remaining variables. The variables set to zero are called *nonbasic*, and the remaining ones are called *basic*. In the profit maximization example, each of the 10 solutions in Table 6-1 is basic, but only A, B, C, D, and E are basic and feasible.

Basic Feasible Solution. A basic solution satisfying the nonnegativity conditions on the variables is called a basic feasible solution. Note that solutions 1, 2, 6, 8, and 9 in Table 6-1 are the basic feasible solutions.

Degenerate Basic Solution. If a basic variable has zero value in a basic solution, the solution is called a degenerate basic solution.

Degenerate Basic Feasible Solution. If a basic variable has zero value in a basic feasible solution, the solution is called degenerate basic feasible solution.

Optimum Solution. A feasible solution minimizing the cost function is called an optimum solution. The point D in Fig. 6-1 corresponds to the optimum solution.

Optimum Basic Solution. It is a basic feasible solution that has optimum cost function value. From Table 6-1 and Fig. 6-1 it is clear that only solution number 8 is the optimum basic solution.

Convex Polyhedron. If the feasible set for an LP problem is bounded, it is called a convex polyhedron. As an example, the polygon ABCDE in Fig. 6-1 represents a convex polyhedron for the problem of Example 6.2.

Basis. Columns of the coefficient matrix **A** in Eq. (6.10) of the constraint equations corresponding to the basic variables are said to form a basis for the m -dimensional vector space. Any other m -dimensional vector can be expressed as a linear combination of the basis vectors.

Example 6.3 presents how to determine basic solutions.

EXAMPLE 6.3 Determination of Basic Solutions

Find all basic solutions for the following problem and identify basic feasible solutions in a figure of the feasible set: maximize $z = 4x_1 + 5x_2$ subject to $-x_1 + x_2 \leq 4$, $x_1 + x_2 \leq 6$, and $x_1, x_2 \geq 0$.

Solution. The feasible region for the problem is shown in Fig. 6-2. Introducing slack variables x_3 and x_4 into the constraint equations and converting maximization of z to minimization, the problem is written in the standard LP form as

$$\text{minimize } f = -4x_1 - 5x_2 \quad (\text{a})$$

subject to

$$-x_1 + x_2 + x_3 = 4 \quad (\text{b})$$

$$x_1 + x_2 + x_4 = 6 \quad (\text{c})$$

$$x_i \geq 0; \quad i = 1 \text{ to } 4 \quad (\text{d})$$

Since there are four variables and two constraints in Eqs. (b) and (c) ($n = 4$, $m = 2$), the problem has six basic solutions; i.e., there are six different ways in which two

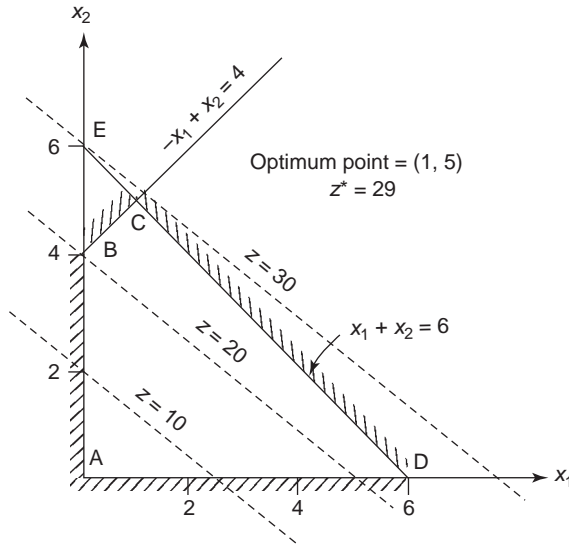


FIGURE 6-2 Graphical solution for the LP problem of Example 6.3. Optimum point = (1, 5). $z^* = 29$.

TABLE 6-2 Basic Solutions for Example 6-3

No.	x_1	x_2	x_3	x_4	f	Location in Fig. 6-2
1	0	0	4	6	0	A
2	0	4	0	2	-20	B
3	0	6	-2	0	—	infeasible
4	-4	0	0	10	—	infeasible
5	6	0	10	0	-24	D
6	1	5	0	0	-29	C

of the variables can be chosen as nonbasic. These solutions are obtained from Eqs. (b) and (c) by choosing two variables as nonbasic and the remaining two as basic. For example, x_1 and x_2 may be chosen as nonbasic, i.e., $x_1 = 0$, $x_2 = 0$. Then Eqs. (b) and (c) give $x_3 = 4$, $x_4 = 6$. Also with $x_1 = 0$ and $x_3 = 0$, Eqs. (b) and (c) give $x_2 = 2$ and $x_4 = 2$ as another basic solution. Similarly, the remaining basic solutions are obtained by selecting two variables as nonbasic (zero) and solving for the other two from Eqs. (b) and (c). The six basic solutions for the problem are summarized in Table 6-2 along with the corresponding cost function values. The basic feasible solutions are 1, 2, 5, and 6. These correspond to points (0, 0), (0, 4), (6, 0), and (1, 5) in Fig. 6-2, respectively. The minimum value of the cost function is obtained at the point (1, 5) as $f = -29$ (maximum value of $z = 29$). In Section 6.3, we shall introduce a systematic tabular procedure based on the Gaussian elimination method of Sections B.3 and B.4 to determine all the basic solutions of Eqs. (b) and (c).

6.2.3 Optimum Solution for LP Problems

Now some important theorems that define the optimum solution for LP problems are stated and explained.

Theorem 6.1 Extreme Points and Basic Feasible Solutions The collection of feasible solutions for an LP problem constitutes a convex set whose extreme points correspond to basic feasible solutions.

This theorem relates *extreme points of the convex polyhedron to the basic feasible solutions*. This is an important result giving geometric meaning to the basic feasible solutions; they are the vertices of the polyhedron representing the feasible set for an LP problem. As an example, basic feasible solutions in Table 6-1 correspond to vertices of the feasible set in Fig. 6-1. *Theorem 6.2 establishes the importance of the basic feasible solutions.*

Theorem 6.2 Basic Theorem of Linear Programming Let the $m \times n$ coefficient matrix \mathbf{A} of the constraint equations have full row rank, i.e., $\text{rank}(\mathbf{A}) = m$. Then

1. If there is a feasible solution, there is a basic feasible solution,
2. If there is an optimum feasible solution, there is an optimum basic feasible solution.

Part 1 of the theorem says that if there is any feasible solution to the LP problem, then there must be at least one extreme point or vertex of the *convex feasible set*. Part 2 of the theorem says that if the LP problem has a solution, then it is at least at one of the vertices of the *convex polyhedron* representing feasible solutions. There can be multiple optimum solutions if the cost function is parallel to one of the active constraints, as we have seen before in Chapter 3. As noted earlier, the LP problem has an infinite number of feasible designs. We seek a feasible design that minimizes the cost function. Theorem 6.2 says that such a solution must be one of the basic feasible solutions, i.e., at one of the extreme points of the convex feasible set. Thus, our task of solving an LP problem has been reduced to the search for an optimum only among the basic feasible solutions. For a problem having n variables and m constraints, the maximum number of basic solutions is obtained by counting the total number of combinations where m variables are nonzero out of a total of n variables. This number is given by the formula:

$$\# \text{ of combinations} = \binom{n}{m} = \frac{n!}{m!(n-m)!} \quad (6.12)$$

This formula gives only a finite number of basic solutions. Thus according to Theorem 6.2, the optimum solution is at one of these points that is also feasible. We need to search these solutions systematically for the optimum. *The Simplex method of the next section is based on searching among the basic feasible solutions to reduce the cost function continuously until an optimum point is reached.*

6.3 Basic Ideas and Steps of the Simplex Method

Basics of the Simplex method for solving LP problems are described. Ideas of a canonical form, pivot row, pivot column, pivot element, and pivot step are introduced. The Simplex tableau is introduced and its notation is explained. The method is described as an extension of the standard Gauss-Jordan elimination process for solving a system of linear equations $\mathbf{Ax} = \mathbf{b}$, where \mathbf{A} is an $m \times n$ ($m < n$) matrix, \mathbf{x} is an n -vector, and $\mathbf{b} \geq \mathbf{0}$ is an m -vector. In this section, the Simplex method is developed and illustrated for “ \leq type” constraints since with

such constraints, the method can be developed in a straightforward manner. In the next section, “ \geq type” and equality constraints that require special treatment in the Simplex method are discussed. A detailed derivation of the Simplex method is presented in Chapter 7.

Theorem 6.2 guarantees that one of the basic feasible solutions is an optimum for the LP problem. The basic idea of the Simplex method is simply to proceed from one basic feasible solution to another in a way that continually decreases the cost function until the minimum is reached. Thus, to solve an LP problem we need a method to systematically find basic feasible solutions of the linear system of equations in Eqs. (6.8). The *Gauss-Jordan elimination process* provides such a procedure (for a review of the method, see Appendix B, Sections B.3 and B.4). Before the Simplex method is developed, the idea of *Simplex, canonical form tableau*, and *pivot step* are explained. These are fundamental in the development of the method.

6.3.1 The Simplex

A *Simplex* in two-dimensional space is formed by any three points that do not lie on a straight line. In three-dimensional space, it is formed by four points that do not lie in the same plane. Three points can lie in a plane, and the fourth one has to lie outside the plane. In general, a Simplex in the n -dimensional space is a convex hull of any $(n + 1)$ points that do not lie on one hyperplane. A *convex hull* of $(n + 1)$ points is the smallest convex set containing all the points. Thus, the *Simplex represents a convex set*.

6.3.2 Canonical Form/General Solution of $\mathbf{Ax} = \mathbf{b}$

The idea of a canonical form is important in the development of the Simplex method. Therefore, we introduce this idea and discuss its use. An $m \times n$ system of simultaneous equations given in Eq. (6.10) with $\text{rank}(\mathbf{A}) = m$ is said to be in the *canonical form* if each equation has a variable (with unit coefficient) that does not appear in any other equation. A canonical form in general is written as follows:

$$\begin{aligned} x_1 + a_{1,m+1}x_{m+1} + a_{1,m+2}x_{m+2} + \dots + a_{1,n}x_n &= b_1 \\ x_2 + a_{2,m+1}x_{m+1} + a_{2,m+2}x_{m+2} + \dots + a_{2,n}x_n &= b_2 \\ \cdot & \\ \cdot & \\ \cdot & \\ x_m + a_{m,m+1}x_{m+1} + a_{m,m+2}x_{m+2} + \dots + a_{m,n}x_n &= b_m \end{aligned} \tag{6.13}$$

Note that variables x_1 to x_m appear in only one of the equations; x_1 appears in the first equation, x_2 in the second equation, and so on. Note also that this sequence of variables x_1 to x_m in Eq. (6.13) is chosen only for convenience. In general, any of the variables x_1 to x_n may be associated with the first equation as long as it does not appear in any other equation. Similarly, the second equation need not be associated with the second variable x_2 . This will become clearer when we discuss the Simplex method.

The Gauss-Jordan elimination process can be used to convert a given system of equations into the canonical form of Eq. (6.13). It is also possible to write the canonical form of Eq. (6.13) as a matrix equation, as also explained in Section B.4 of Appendix B:

$$\mathbf{I}_{(m)}\mathbf{x}_{(m)} + \mathbf{Q}\mathbf{x}_{(n-m)} = \mathbf{b} \tag{6.14}$$

where

$$\begin{aligned} \mathbf{I}_{(m)} &= m\text{-dimensional identity matrix} \\ \mathbf{x}_{(m)} &= [x_1 \ x_2 \ \dots \ x_m]^T; \text{ vector of dimension } m \\ \mathbf{x}_{(n-m)} &= [x_{m+1} \ \dots \ x_n]^T; \text{ vector of dimension } (n - m) \end{aligned}$$

$$\mathbf{Q} = m \times (n - m) \text{ matrix consisting of coefficients of the variables } x_{m+1} \text{ to } x_n \text{ in Eq. (6.13)}$$

$$\mathbf{b} = [b_1 \ b_2 \ \dots \ b_m]^T; \text{ vector of dimension } m$$

Basic and Nonbasic Variables The canonical form in Eq. (6.13) or Eq. (6.14) gives a *general solution* for $\mathbf{Ax} = \mathbf{b}$ as $\mathbf{x}_{(m)} = \mathbf{b} - \mathbf{Qx}_{(n-m)}$ (Appendix B, Section B.4). It is seen that $\mathbf{x}_{(n-m)}$ can be assigned different values and the corresponding values for $\mathbf{x}_{(m)}$ can be calculated from this equation. Thus $\mathbf{x}_{(m)}$ are dependent variables and $\mathbf{x}_{(n-m)}$ are independent variables. A *particular solution* of the equations is obtained if we set the independent variables to zero, $\mathbf{x}_{(n-m)} = \mathbf{0}$. Then from Eq. (6.13) or (6.14), $\mathbf{x}_{(m)} = \mathbf{b}$. The variables set to zero in $\mathbf{x}_{(n-m)}$ are called *nonbasic*, and the variables $\mathbf{x}_{(m)}$ solved from Eq. (6.13) or (6.14) are called *basic*. The solution thus obtained is called a *basic solution*. If the right side parameters b_i are ≥ 0 , then the canonical form gives a *basic feasible solution*.

Equations (g) to (i) in Example 6.2 represent a canonical form. In these equations, the variables x_1 and x_2 are nonbasic, so they have zero value. The variables x_3, x_4 , and x_5 are basic and their values are readily obtained from the canonical form as $x_3 = 16, x_4 = 1$, and $x_5 = 1$. Similarly, Eqs. (b) and (c) of Example 6.3 represent a canonical form giving a basic solution of $x_1 = 0, x_2 = 0, x_3 = 4, x_4 = 6$.

6.3.3 Tableau

It is customary to represent the canonical form in a tableau as shown in Table 6-3. A *tableau* is defined as the representation of a scene or a picture. It is a convenient way of representing all the necessary information related to an LP problem. With the Simplex method, the tableau consists of coefficients of the design variables in the cost and constraint functions. The tableau in Table 6-3 does not contain coefficients of the cost function; they can, however, be included, as we shall see later.

It is important to understand the structure and notation of the tableau as explained in the following because the tableau is used later to develop the Simplex method:

1. *The entries of the tableau are obtained by reducing the linear system of equations $\mathbf{Ax} = \mathbf{b}$ to the canonical form of Eq. (6.14).* In Table 6-3, the first m columns correspond to the identity matrix, the next $(n - m)$ columns correspond to the \mathbf{Q} matrix, and the last column corresponds to the vector \mathbf{b} on the right side (RS) of Eq. (6.14).
2. *Each column of the tableau is associated with a variable; x_1 with the first column, x_2 with the second, and so on.* This is because the i th column contains the coefficient of the variable x_i in each of the rows in Eq. (6.14).
3. *Each row of the tableau contains coefficients of the corresponding row in Eq. (6.13) or (6.14).*
4. *Each row of the tableau is also associated with a variable as indicated in the column named "Basic" on the left in Table 6-3.* These variables correspond to the columns of the identity matrix in the tableau. In Table 6-3, x_1 corresponds to the first column,

TABLE 6-3 Representation of a Canonical Form in a Tableau

#	Basic↓	x_1	x_2	•	•	•	x_m	x_{m+1}	x_{m+2}	•	•	•	x_n	RS
1	x_1	1	0	•	•	•	0	$a_{1,m+1}$	$a_{1,m+2}$	•	•	•	$a_{1,n}$	b_1
2	x_2	0	1	•	•	•	0	$a_{2,m+1}$	$a_{2,m+2}$	•	•	•	$a_{2,n}$	b_2
3	x_3	0	0	•	•	•	0	$a_{3,m+1}$	$a_{3,m+2}$	•	•	•	$a_{3,n}$	b_3
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
m	x_m	0	0	•	•	•	1	$a_{m,m+1}$	$a_{m,m+2}$	•	•	•	$a_{m,n}$	b_m

x_2 to the second column, and so on. Note, however, that columns of the identity matrix can appear anywhere in the tableau. They need not be in any sequence either. Since variables associated with the identity matrix in Eq. (6.14) are basic, the leftmost column then identifies a basic variable associated with each row. This will become clearer later when we solve example problems.

5. *Since each basic variable appears in only one row, its value is immediately available in the rightmost column* (recall that by definition, the nonbasic variables have zero value). For the example of Table 6-3, the basic variables have the values $x_i = b_i$, $i = 1$ to m . If all $b_i \geq 0$, we have a basic feasible solution.
6. *The tableau identifies basic and nonbasic variables, and gives their values, i.e., it gives a basic solution.* We shall see later that the tableau can be augmented with the cost function expression, and in that case, it will also immediately give the value of the cost function associated with the basic solution.
7. Columns associated with the basic variables are called *basic columns* and others are called *nonbasic columns*.

Example 6.4 describes the canonical form and tableau.

EXAMPLE 6.4 Canonical Form and Tableau

Write the canonical form of Example 6.2 in a tableau.

Solution. Table 6-4 shows Eq. (f) of Example 6.2 written in the notation of the tableau of Table 6-3. Note that for this example, the number of equations is three and the number of variables is five, i.e., $m = 3$ and $n = 5$.

The variables x_3 , x_4 , and x_5 appear in one and only one equation, so the columns x_3 , x_4 , and x_5 define the identity matrix $\mathbf{I}_{(m)}$ of the canonical form of Eq. (6.14). The basic and nonbasic variable vectors $\mathbf{x}_{(m)}$ and $\mathbf{x}_{(n-m)}$ are defined as

$$\text{Basic: } \mathbf{x}_{(m)} = [x_3 \ x_4 \ x_5]^T; \quad \text{Nonbasic: } \mathbf{x}_{(n-m)} = [x_1 \ x_2]^T \quad (\text{a})$$

The matrix \mathbf{Q} of Eq. (6.14) is identified as

$$\mathbf{Q} = \begin{bmatrix} 1 & 1 \\ \frac{1}{28} & \frac{1}{14} \\ \frac{1}{14} & \frac{1}{24} \end{bmatrix} \quad (\text{b})$$

If x_1 and x_2 are taken as nonbasic, then the values for the basic variables are obtained from the tableau as $x_3 = 16$, $x_4 = 1$, $x_5 = 1$.

TABLE 6-4 Tableau for LP Problem of Example 6.4

<i>Basic</i> ↓		x_1	x_2	x_3	x_4	x_5	b
1	x_3	1	1	1	0	0	16
2	x_4	$\frac{1}{28}$	$\frac{1}{14}$	0	1	0	1
3	x_5	$\frac{1}{14}$	$\frac{1}{24}$	0	0	1	1

6.3.4 The Pivot Step

In the Simplex method, we want to systematically search among the basic feasible solutions for the optimum design. *We must have a basic feasible solution to initiate the Simplex method.* Starting from the basic feasible solution, we want to find another that decreases the cost function. This can be done by interchanging a current basic variable with a nonbasic variable. That is, a current basic variable is made nonbasic (i.e., reduced to 0 from a positive value) and a current nonbasic variable is made basic (i.e., increased from 0 to a positive value). The *pivot step* accomplishes this task and results in a new canonical form (general solution), as explained in the following.

Let us select a basic variable x_p ($1 \leq p \leq m$) to be replaced by a nonbasic variable x_q for $(n - m) \leq q \leq n$. We will describe later how to determine x_p and x_q . The p th basic column is to be interchanged with the q th nonbasic column. This is possible only when the element in the p th column and q th row is nonzero; i.e., $a_{pq} \neq 0$. The element $a_{pq} \neq 0$ is called the *pivot element*. *The pivot element must always be positive* in the Simplex method as we shall see later. Note that x_q will be basic if it is eliminated from all the equations except the p th one. This can be accomplished by performing a *Gauss-Jordan elimination step* on the q th column of the tableau shown in Table 6-3 using the p th row for elimination. This will give $a_{pq} = 1$ and zeros elsewhere in the q th column. The row used for the elimination process (p th row) is called the *pivot row*. The column on which the elimination is performed (q th column) is called the *pivot column*. The process of interchanging one basic variable with a nonbasic variable is called the *pivot step*.

Let a'_{ij} denote the new coefficients in the canonical form after the pivot step. Then, the *pivot step* for performing elimination in the q th column using the p th row as the pivot row is described by the following general equations.

Divide the pivot row (p) by the pivot element a_{pq} :

$$a'_{pj} = a_{pj} / a_{pq} \text{ for } j = 1 \text{ to } n; \quad b'_p = b_p / a_{pq} \quad (6.15)$$

Eliminate x_q from all rows except the p th row:

$$a'_{ij} = a_{ij} - (a_{pj} / a_{pq}) a_{iq}; \quad \begin{cases} i \neq p, i = 1 \text{ to } m \\ j = 1 \text{ to } n \end{cases} \quad (6.16)$$

$$b'_i = b_i - (b_p / a_{pq}) a_{iq}; \quad i \neq p, i = 1 \text{ to } m \quad (6.17)$$

In Eq. (6.15), the p th row of the tableau is simply divided by the pivot element a_{pq} . Equations (6.16) and (6.17) perform the elimination step in the q th column of the tableau. Elements in the q th column above and below the p th row are reduced to zero by the elimination process thus eliminating x_q from all the rows except the p th row. These equations may be coded into a computer program to perform the pivot step. On completion of the pivot step, a new canonical form for the equation $\mathbf{Ax} = \mathbf{b}$ is obtained; i.e., a new basic solution of the equations is obtained. The process of interchanging roles of two variables is illustrated in Example 6.5.

EXAMPLE 6.5 Pivot Step—Interchange of Basic and Nonbasic Variables

Assuming x_3 and x_4 as basic variables, Example 6.3 is written in the canonical form as follows: minimize $f = -4x_1 - 5x_2$ subject to $-x_1 + x_2 + x_3 = 4$, $x_1 + x_2 + x_4 = 6$, $x_i \geq 0$; $i = 1$ to 4. Obtain a new canonical form by interchanging the roles of x_1 and x_4 , i.e., make x_1 a basic variable and x_4 a nonbasic variable.

TABLE 6-5 Pivot Step to Interchange Basic Variable x_4 with Nonbasic Variable x_1 for Example 6.5

Initial canonical form.

<i>Basic</i> ↓		x_1	x_2	x_3	x_4	b
1	x_3	-1	1	1	0	4
2	x_4	1	1	0	1	6

Basic solution: Nonbasic variables: $x_1 = 0, x_2 = 0$

Basic variables: $x_3 = 4, x_4 = 6$

To interchange x_1 with x_4 , choose row 2 as the pivot row and column 1 as the pivot column. Perform elimination using a_{21} as the pivot element.

Result of the pivot operation: second canonical form.

<i>Basic</i> ↓		x_1	x_2	x_3	x_4	b
1	x_3	0	2	1	1	10
2	x_1	1	1	0	1	6

Basic solution: Nonbasic variables: $x_2 = 0, x_4 = 0$

Basic variables: $x_1 = 6, x_3 = 10$

Solution. The given canonical form can be written in a tableau as shown in Table 6-5; x_1 and x_2 are nonbasic and x_3 and x_4 are basic, i.e., $x_1 = x_2 = 0, x_3 = 4, x_4 = 6$. This corresponds to point A in Fig. 6-2. In the tableau, the basic variables are identified in the leftmost column and the rightmost column gives their values. Also, the basic variables can be identified by examining columns of the tableau. The variables associated with the columns of the identity matrix are basic; e.g., variables x_3 and x_4 in Table 6-5. Location of the positive unit element in a basic column identifies the row whose right side parameter b_i is the current value of the basic variable associated with that column. For example, the basic column x_3 has unit element in the first row, and so x_3 is the basic variable associated with the first row. Similarly, x_4 is the basic variable associated with row 2.

To make x_1 basic and x_4 a nonbasic variable, one would like to make $a'_{21} = 1$ and $a'_{11} = 0$. This will replace x_1 with x_4 as the basic variable and a new canonical form will be obtained. The second row is treated as the pivot row, i.e., $a_{21} = 1$ ($p = 2, q = 1$) is the pivot element. Performing Gauss-Jordan elimination in the first column with $a_{21} = 1$ as the pivot element, we obtain the second canonical form as shown in Table 6-5. For this canonical form, $x_2 = x_4 = 0$ are the nonbasic variables and $x_1 = 6$ and $x_3 = 10$ are the basic variables. Thus, referring to Fig. 6-2, this pivot step results in a move from the extreme point A(0, 0) to an adjacent extreme point D(6, 0).

6.3.5 Basic Steps of the Simplex Method

In this section, we shall illustrate the basic steps of the Simplex method with an example problem. In the next subsection, we shall explain the basis for these steps and summarize them in a step-by-step general algorithm. The method starts with a basic feasible solution, i.e., at a vertex of the convex polyhedron. A move is then made to an adjacent vertex while

maintaining feasibility of the new solution as well as reducing the cost function. This is accomplished by replacing a basic variable with a nonbasic variable. In the Simplex method, movements are to the adjacent vertices only. Since there may be several points adjacent to the current vertex, we naturally wish to choose the one that makes the greatest improvement in the cost function f . If adjacent points make identical improvements in f , the choice becomes arbitrary. An improvement at each step ensures no backtracking. Two basic questions now arise:

1. How to choose a current nonbasic variable that should become basic?
2. Which variable from the current basic set should become nonbasic?

The Simplex method answers these questions based on some theoretical considerations which shall be discussed in Chapter 7. Here, we consider an example to illustrate the basic steps of the Simplex method that answer the foregoing two questions. Before presentation of the example problem, an important requirement of the Simplex method is discussed. In this method the *cost function must always be given in terms of the nonbasic variables only*. To accomplish this, the cost function expression $\mathbf{c}^T \mathbf{x} = f$ is written as another linear equation in the Simplex tableau; for example, the $(m + 1)$ th row. One then performs the pivot step on the entire set of $(m + 1)$ equations so that x_1, x_2, \dots, x_m and f are the basic variables. This way the last row of the tableau representing the cost function expression is automatically given in terms of the nonbasic variables after each pivot step. The coefficients in the nonbasic columns of the last row are called the *reduced cost coefficients* written as c'_j . Example 6.6 describes the steps of the Simplex method in a systematic way.

EXAMPLE 6.6 Steps of the Simplex Method

Solve the following LP problem: maximize

$$z = 2x_1 + x_2 \text{ subject to } 4x_1 + 3x_2 \leq 12, 2x_1 + x_2 \leq 4, x_1 + 2x_2 \leq 4, x_1, x_2 \geq 0$$

Solution. The graphical solution for the problem is given in Fig. 6-3. It can be seen that the problem has an infinite number of solutions along the line C–D ($z^* = 4$) because the objective function is parallel to the second constraint. The Simplex method is illustrated in the following steps:

1. *Convert the problem to the standard form.* We write the problem in the standard LP form by transforming the maximization of z to minimization of $f = -2x_1 - x_2$, and adding slack variables x_3, x_4 , and x_5 to the constraints. Thus, the problem becomes

$$\text{minimize } f = -2x_1 - x_2 \tag{a}$$

subject to

$$4x_1 + 3x_2 + x_3 = 12 \tag{b}$$

$$2x_1 + x_2 + x_4 = 4 \tag{c}$$

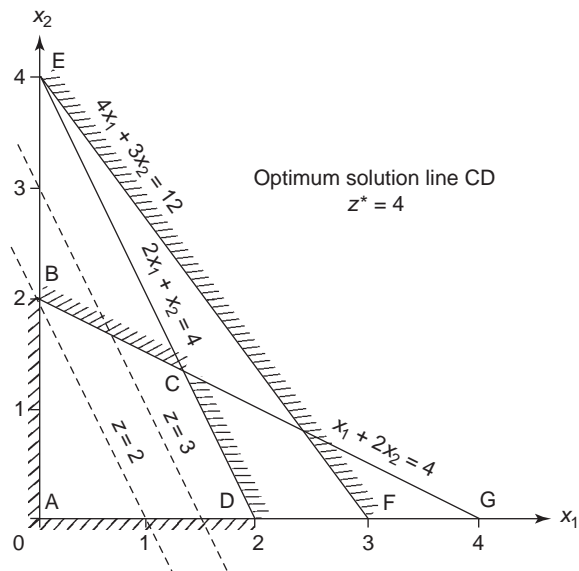


FIGURE 6-3 Graphical solution for the LP problem of Example 6.6. Optimum solution: along line C–D. $z^* = 4$.

$$x_1 + 2x_2 + x_5 = 4 \quad (d)$$

$$x_i \geq 0; \quad i = 1 \text{ to } 5 \quad (e)$$

We use the tableau and notation of Table 6-3 which will be augmented with the cost function expression as the last row. The initial tableau for the problem is shown in Table 6-6, where the cost function expression $-2x_1 - x_2 = f$ is written as the last row. Note also that the cost function is in terms of only the nonbasic variables x_1 and x_2 . *This is one of the basic requirements of the Simplex method—that the cost function always be in terms of the nonbasic variables.* When the cost function is only in terms of the nonbasic variables, then the cost coefficients in the last row are the *reduced cost coefficients*, written as c'_j .

2. *Initial basic feasible solution.*

To initiate the Simplex method, a basic feasible solution is needed. This is already available in Table 6-6 which is given as:

$$\text{basic variables:} \quad x_3 = 12, x_4 = 4, x_5 = 4$$

$$\text{nonbasic variables:} \quad x_1 = 0, x_2 = 0$$

$$\text{cost function:} \quad f = 0$$

Note that the cost row gives $0 = f$ after substituting for x_1 and x_2 . This solution represents point A in Fig. 6-3 where none of the constraints is active except the nonnegativity constraints on the variables.

3. *Optimality check.* We scan the cost row, which should have nonzero entries only in the nonbasic columns, i.e., x_1 and x_2 . *If all the nonzero entries are nonnegative, then we have an optimum solution* because the cost function

TABLE 6-6 Initial Tableau for the LP Problem of Example 6.6

Basic↓		x_1	x_2	x_3	x_4	x_5	b
1	x_3	4	3	1	0	0	12
2	x_4	2	1	0	1	0	4
3	x_5	1	2	0	0	1	4
Cost function		-2	-1	0	0	0	<i>f</i>

Notation. The reduced cost coefficients in the nonbasic columns are boldfaced. The selected negative reduced cost coefficient is boxed.

TABLE 6-7 Selection of Pivot Column and Pivot Row for Example 6.6

Basic↓		x_1	x_2	x_3	x_4	x_5	b	Ratio: b_i/a_{i1} ; $a_{i1} > 0$
1	x_3	4	3	1	0	0	12	$\frac{12}{4} = 3$
2	x_4	2	1	0	1	0	4	$\frac{4}{2} = 2 \leftarrow$ smallest
3	x_5	1	2	0	0	1	4	$\frac{4}{1} = 4$
Cost function		-2	-1	0	0	0	<i>f</i>	

The selected pivot element is boxed. Selected pivot row and column are shaded. x_1 should become basic (pivot column). x_4 row has the smallest ratio, and so x_4 should become nonbasic.

cannot be reduced any further and the Simplex method is terminated. There are negative entries in the cost row so the current basic feasible solution is *not optimum* (see Chapter 7 for further explanation).

4. *Choice of a nonbasic variable to become basic.* We select a nonbasic column having a negative cost coefficient; i.e., -2 in the x_1 column. This identifies a nonbasic variable (x_1) that should become basic. Thus, eliminations will be performed in the x_1 column. This answers question 1 posed earlier: “How to choose a current nonbasic variable that should become basic?” Note also that when there is more than one negative entry in the cost row, the variable tapped to become basic is arbitrary among the indicated possibilities. The usual convention is to select a variable associated with the smallest value in the cost row (or, negative element with the largest absolute value).

Notation. The boxed negative reduced cost coefficient in Table 6-6 indicates the nonbasic variable associated with that column selected to become basic, a notation that is used throughout.

5. *Selection of a basic variable to become nonbasic.* To identify which current basic variable should become nonbasic (i.e., to select the pivot row), we take ratios of the right side parameters with the positive elements in the x_1 column as shown in Table 6-7. We identify the row having the smallest positive ratio, i.e., the second row. This will make x_4 nonbasic. The pivot element is $a_{21} = 2$ (the intersection of pivot row and pivot column). This answers the question 2 posed earlier: “Which variable from the current basic set should become nonbasic?” Selection of the row with the smallest ratio as the pivot row maintains feasibility of the new basic solution. This is justified in Chapter 7.

TABLE 6-8 Second Tableau for Example 6.6 Making x_1 a Basic Variable

	<i>Basic</i> ↓	x_1	x_2	x_3	x_4	x_5	b
1	x_3	0	1	1	-2	0	4
2	x_1	1	0.5	0	0.5	0	2
3	x_5	0	1.5	0	-0.5	1	2
Cost function		0	0	0	1	0	$f + 4$

The cost coefficient in nonbasic columns are nonnegative; the tableau gives the optimum solution.

Notation. The selected pivot element is also boxed, and the pivot column and row are shaded throughout.

6. *Pivot operation.* We perform eliminations in column x_1 using row 2 as the pivot row and Eqs. (6.15) to (6.17) to eliminate x_1 from rows 1, 3, and the cost row as follows:

- divide row 2 by 2, the pivot element
- multiply new row 2 by 4 and subtract from row 1 to eliminate x_1 from row 1
- subtract new row 2 from row 3 to eliminate x_1 from row 3
- multiply new row 2 by 2 and add to the cost row to eliminate x_1

As a result of this elimination step, a new tableau is obtained as shown in Table 6-8. The new basic feasible solution is given as

$$\text{basic variables: } x_3 = 4, x_1 = 2, x_5 = 2$$

$$\text{nonbasic variables: } x_2 = 0, x_4 = 0$$

$$\text{cost function: } 0 = f + 4, f = -4$$

7. This solution is identified as point D in Fig. 6-3. We see that the cost function has been reduced from 0 to -4 . All coefficients in the last row are nonnegative so no further reduction of the cost function is possible. Thus, the foregoing solution is the optimum. Note that for this example, only one iteration of the Simplex method gave the optimum solution. In general, more iterations are needed until all coefficients in the cost row become nonnegative.

Note that the cost coefficients corresponding to the nonbasic variable x_2 in the last row is zero in the final tableau. This is an indication of *multiple solutions* for the problem. In general, when the reduced cost coefficient in the last row corresponding to a nonbasic variable is zero, the problem may have multiple solutions. We shall discuss this point later in more detail.

Let us see what happens if we do not select a row with the least ratio as the pivot row. Let $a_{31} = 1$ in the third row be the pivot element in Table 6-6. This will interchange nonbasic variable x_1 with the basic variable x_5 . Performing the elimination steps in the first column as explained earlier, we obtain the new tableau given in Table 6-9. From the tableau, we have

$$\text{basic variables: } x_3 = -4, x_4 = -4, x_1 = 4$$

$$\text{nonbasic variables: } x_2 = 0, x_5 = 0$$

$$\text{cost function: } 0 = f + 8, f = -8$$

TABLE 6-9 Result of Improper Pivoting in Simplex Method for LP Problem of Example 6.6

	Basic↓	x_1	x_2	x_3	x_4	x_5	b
1	x_3	0	-5	1	0	-4	-4
2	x_4	0	-3	0	1	-2	-4
3	x_1	1	2	0	0	1	4
Cost function		0	3	0	0	4	$f + 8$

The pivot step making x_1 basic and x_5 nonbasic in Table 6-6 gives a basic solution that is not feasible.

The foregoing solution corresponds to point G in Fig. 6-3. We see that this basic solution is not feasible because x_3 and x_4 have negative values. Thus, we *conclude that if a row with the smallest ratio (of right sides with positive elements in the pivot column) is not selected, the new basic solution is not feasible.* Note that a spreadsheet program, such as Excel, can be used to carry out the pivot step. Such a program can facilitate learning of the Simplex method without getting bogged down with the manual elimination process.

6.3.6 Simplex Algorithm

In the previous subsection, basic steps of the Simplex method are explained and illustrated with an example problem. In this subsection, the underlying principles for these steps are summarized in two theorems, called basic theorems of linear programming. We have seen that in general the reduced cost coefficients c'_j of the nonbasic variables may be positive, negative, or zero. Let one of c'_j be negative, then if a positive value is assigned to the associated nonbasic variable (i.e., it is made basic), the value of f will decrease. If more than one negative c'_j is present, a widely used rule of thumb is to choose the nonbasic variable associated with the smallest c'_j (i.e., the most negative c'_j) to become basic. Thus, if any c'_j for $(m + 1) \leq j \leq n$ (for nonbasic variables) is negative, then it is possible to find a new basic feasible solution (if one exists) that will further reduce the cost function. If a c'_j is zero, then the associated nonbasic variable can be made basic without affecting the cost function value. If all c'_j are nonnegative, then it is not possible to reduce the cost function any further, and the current basic feasible solution is optimum. These ideas are summarized in the following Theorems 6.3 and 6.4.

Theorem 6.3 Improvement of Basic Feasible Solution Given a nondegenerate basic feasible solution with the corresponding cost function f_0 , suppose that $c'_j < 0$ for some j . Then, there is a feasible solution with $f < f_0$. *If the j th nonbasic column associated with c'_j can be substituted for some column in the original basis, the new basic feasible solution will have $f < f_0$.* If the j th column cannot be substituted to yield a basic feasible solution (i.e., there is no positive element in the j th column), then the *feasible set is unbounded* and the cost function can be made arbitrarily small (toward negative infinity).

Theorem 6.4 Optimum Solution for LP Problems If a basic feasible solution has reduced cost coefficients $c'_j \geq 0$ for all j , then it is optimum.

According to Theorem 6.3, the basic procedure of the Simplex method is to start with an initial basic feasible solution, i.e., at the vertex of the convex polyhedron. If this solution is

not optimum according to Theorem 6.4, then a move is made to an adjacent vertex to reduce the cost function. The procedure is continued until the optimum is reached. *The steps of the Simplex method illustrated in the previous subsection in Example 6.6 are summarized as follows assuming that the initial tableau has been set up as described earlier:*

Step 1. Initial Basic Feasible Solution This is readily obtained if all constraints are “ \leq type” because the slack variables can be selected as basic and the real variables as nonbasic. If there are equality or “ \geq type” constraints, then the two-phase simplex procedure explained in the next section must be used.

Step 2. The Cost Function Must be in Terms of Only the Nonbasic Variables This is readily available when there are only “ \leq type” constraints and slack variables are added into them to convert the inequalities to equalities. The slack variables are basic, and they do not appear in the cost function. In subsequent iterations, eliminations must also be performed in the cost row.

Step 3. If All the Reduced Cost Coefficients for Nonbasic Variables Are Nonnegative (≥ 0), We Have the Optimum Solution Otherwise, there is a possibility of improving the cost function. We need to select a nonbasic variable that should become basic. We identify a column having negative reduced cost coefficient because the nonbasic variable associated with this column can become basic to reduce the cost function from its current value. This is called the *pivot column*.

Step 4. If All Elements in the Pivot Column Are Negative, Then We Have an Unbounded Problem Design problem formulation should be examined to correct the situation. If there are positive elements in the pivot column, then we take ratios of the right side parameters with the positive elements in the pivot column and identify a row with the smallest positive ratio. In the case of a tie, any row among the tying ratios can be selected. The basic variable associated with this row should become nonbasic (i.e., become zero). The selected row is called the *pivot row*, and its intersection with the pivot column identifies the *pivot element*.

Step 5. Complete the Pivot Step Use the Gauss-Jordan elimination procedure and the pivot row identified in Step 4. *Elimination must also be performed in the cost function row* so that it is only in terms of nonbasic variables in the next tableau. This step eliminates the nonbasic variable identified in Step 3 from all the rows except the pivot row.

Step 6. Identify Basic and Nonbasic Variables, and Their Values Identify the cost function value and go to Step 3.

Note that when all the reduced cost coefficient c'_j in the nonbasic columns are strictly positive, the optimum solution is unique. If at least one c'_j is zero in a nonbasic column, then there is a possibility of an alternate optimum. If the nonbasic variable associated with a zero reduced cost coefficient can be made basic by using the foregoing procedure, the extreme point (vertex) corresponding to an alternate optimum is obtained. Since the reduced cost coefficient is zero, the optimum cost function value will not change. Any point on the line segment joining the optimum extreme points also corresponds to an optimum. Note that all these optima are global as opposed to local, although there is *no distinct global optimum*. Geometrically, *multiple optima* for an LP problem imply that the cost function hyperplane is parallel to one of the constraint hyperplanes. Example 6.7 shows how to obtain a solution for an LP problem using the Simplex method.

EXAMPLE 6.7 Solution by the Simplex Method

Using the Simplex method, find the optimum (if one exists) for the LP problem of Example 6.3:

$$\text{minimize } f = -4x_1 - 5x_2 \quad (\text{a})$$

subject to

$$-x_1 + x_2 + x_3 = 4 \quad (\text{b})$$

$$x_1 + x_2 + x_4 = 6 \quad (\text{c})$$

$$x_i \geq 0; \quad i = 1 \text{ to } 4 \quad (\text{d})$$

Solution. Writing the problem in the Simplex tableau, we obtain the initial canonical form as shown in Table 6-10. From the initial tableau, the basic feasible solution is

basic variables: $x_3 = 4, x_4 = 6$

nonbasic variables: $x_1 = x_2 = 0$

cost function: $f = 0$ from the last row of the tableau

Note that the cost function in the last row is in terms of only nonbasic variables x_1 and x_2 . Thus, coefficients in the x_1 and x_2 columns and the last row are the reduced

TABLE 6-10 Solution of Example 6.7 by the Simplex Method

Initial tableau: x_3 is identified to be replaced with x_2 in the basic set.						
Basic↓	x_1	x_2	x_3	x_4	b	Ratio: b_i/a_{iq}
x_3	-1	1	1	0	4	$\frac{4}{1} = 4 \leftarrow$ smallest
x_4	1	1	0	1	6	$\frac{6}{1} = 6$
Cost	-4	-5	0	0	f	
Second tableau: x_4 is identified to be replaced with x_1 in the basic set.						
Basic↓	x_1	x_2	x_3	x_4	b	Ratio: b_i/a_{iq}
x_2	-1	1	1	0	4	Negative
x_4	2	0	-1	1	2	$\frac{2}{2} = 1$
Cost	-9	0	5	0	$f + 20$	
Third tableau: Reduced cost coefficients in nonbasic columns are nonnegative; the tableau gives optimum point.						
Basic↓	x_1	x_2	x_3	x_4	b	Ratio: b_i/a_{iq}
x_2	0	1	$\frac{1}{2}$	$\frac{1}{2}$	5	Not needed
x_1	1	0	$-\frac{1}{2}$	$\frac{1}{2}$	1	Not needed
Cost	0	0	$\frac{1}{2}$	$\frac{9}{2}$	$f + 29$	

cost coefficients c'_j . Scanning the last row, we observe that there are negative coefficients. Therefore, the current basic solution is not optimum. In the last row, the most negative coefficient of -5 corresponds to the second column. Therefore, we select x_2 to become a basic variable, i.e., elimination should be performed in the x_2 column. This fixes the column index q to 2 in Eq. (6.15). Now taking the ratios of the right side parameters with positive coefficients in the second column b_i/a_{i2} , we obtain a minimum ratio for the first row as 4. This identifies the first row as the pivot row according to Step 4. Therefore, the current basic variable associated with the first row, x_3 , should become nonbasic. Now performing the pivot step on column 2 with a_{12} as the pivot element, we obtain the second canonical form (tableau) as shown in Table 6-10. For this canonical form the basic feasible solution is

$$\begin{aligned} \text{basic variables:} \quad & x_2 = 4, x_4 = 2 \\ \text{nonbasic variables:} \quad & x_1 = x_3 = 0 \end{aligned}$$

The cost function is $f = -20$ ($0 = f + 20$), which is an improvement from $f = 0$. Thus, this pivot step results in a move from $(0, 0)$ to $(0, 4)$ on the convex polyhedron of Fig. 6-2.

The reduced cost coefficient corresponding to the nonbasic column x_1 is negative. Therefore, the cost function can be reduced further. Repeating the above-mentioned process for the second tableau, we obtain $a_{21} = 2$ as the pivot element, implying that x_1 should become basic and x_4 should become nonbasic. The third canonical form is shown in Table 6-10. For this tableau, all the reduced cost coefficients c'_j (corresponding to the nonbasic variables) in the last row are ≥ 0 . Therefore, the tableau yields the optimum solution as $x_1 = 1, x_2 = 5, x_3 = 0, x_4 = 0, f = -29$ ($f + 29 = 0$), which corresponds to the point C $(1, 5)$ in Fig. 6-2.

Example 6.8 demonstrates the solution of the profit maximization problem by the Simplex method.

EXAMPLE 6.8 Solution of Profit Maximization Problem by the Simplex Method

Use the Simplex method to find the optimum solution for the profit maximization problem of Example 6.2.

Solution. Introducing slack variables in the constraints of Eqs. (c) through (e) in Example 6.2, we get the LP problem in the standard form:

$$\text{minimize } f = -400x_1 - 600x_2 \tag{a}$$

subject to

$$x_1 + x_2 + x_3 = 16 \tag{b}$$

$$\frac{1}{28}x_1 + \frac{1}{14}x_2 + x_4 = 1 \tag{c}$$

TABLE 6-11 Solution of Example 6.8 by the Simplex Method

Initial tableau: x_4 is identified to be replaced with x_2 in the basic set.							
<i>Basic</i> ↓	x_1	x_2	x_3	x_4	x_5	b	<i>Ratio: b_i/a_{iq}</i>
x_3	1	1	1	0	0	16	$\frac{16}{1} = 16$
x_4	$\frac{1}{28}$	$\frac{1}{14}$	0	1	0	1	$\frac{1}{1/14} = 14 \leftarrow$ smallest
x_5	$\frac{1}{14}$	$\frac{1}{24}$	0	0	1	1	$\frac{1}{1/24} = 24$
Cost	-400	-600	0	0	0	$f - 0$	

Second tableau: x_3 is identified to be replaced with x_1 in the basic set.							
<i>Basic</i> ↓	x_1	x_2	x_3	x_4	x_5	b	<i>Ratio: b_i/a_{iq}</i>
x_3	$\frac{1}{2}$	0	1	-14	0	2	$\frac{2}{1/2} = 4 \leftarrow$ smallest
x_2	$\frac{1}{2}$	1	0	14	0	14	$\frac{14}{1/2} = 28$
x_5	$\frac{17}{336}$	0	0	$-\frac{7}{12}$	1	$\frac{5}{12}$	$\frac{5/12}{17/336} = \frac{140}{17}$
Cost	-100	0	0	8400	0	$f + 8400$	

Third tableau: Reduced cost coefficients in the nonbasic columns are nonnegative; the tableau gives optimum solution							
<i>Basic</i> ↓	x_1	x_2	x_3	x_4	x_5	b	<i>Ratio: b_i/a_{iq}</i>
x_3	1	0	2	-28	0	4	Not needed
x_2	0	1	-1	28	0	12	Not needed
x_5	0	0	$-\frac{17}{168}$	$\frac{5}{6}$	1	$\frac{3}{14}$	Not needed
Cost	0	0	200	5600	0	$f + 8800$	

$$\frac{1}{14}x_1 + \frac{1}{24}x_2 + x_5 = 1 \tag{d}$$

$$x_i \geq 0; \quad i = 1 \text{ to } 5 \tag{e}$$

Now writing the problem in the standard Simplex tableau, we obtain the initial canonical form as shown in Table 6-11. Thus the initial basic feasible solution is $x_1 = 0$, $x_2 = 0$, $x_3 = 16$, $x_4 = x_5 = 1$, $f = 0$, which corresponds to point A in Fig. 6-1. The initial cost function is zero, and x_3 , x_4 , and x_5 are the basic variables.

Using the Simplex procedure, we note that $a_{22} = \frac{1}{14}$ is the pivot element. This implies that x_4 should be replaced by x_2 in the basic set. Carrying out the pivot operation using the second row as the pivot row, we obtain the second tableau (canonical form) shown in Table 6-11. At this point the basic feasible solution is $x_1 = 0$, $x_2 = 14$, $x_3 = 2$, $x_4 = 0$, $x_5 = \frac{5}{12}$, which corresponds to point E in Fig. 6-1. The cost function is reduced to -8400. The pivot element for the next step is a_{11} , implying that x_3 should be replaced by x_1 in the basic set. Carrying out the pivot operation, we obtain the third canonical form shown in Table 6-11. At this point all reduced cost coefficients (corresponding to nonbasic variables) are nonnegative, so according to Theorem 6.4, we have the optimum solution: $x_1 = 4$, $x_2 = 12$, $x_3 = 0$, $x_5 = \frac{3}{14}$. This corresponds to the D in

Fig. 6-1. The optimum value of the cost function is -8800 . Note that c'_j , corresponding to the nonbasic variables x_3 and x_4 , are positive. Therefore, the global optimum is unique, as may be observed in Fig. 6-1 as well.

Problem in Example 6.9 has multiple solution. The example illustrates how to recognize such solutions with the Simplex method.

EXAMPLE 6.9 LP Problem with Multiple Solutions

Solve the following problem by the Simplex method: maximize $z = x_1 + 0.5x_2$ subject to $2x_1 + 3x_2 \leq 12$, $2x_1 + x_2 \leq 8$, $x_1, x_2 \geq 0$.

Solution. The problem was solved graphically in Section 3.4 of Chapter 3. It has *multiple solutions* as may be seen in Fig. 3-7. We will solve the problem using the Simplex method and discuss how multiple solutions can be recognized for general LP problems. The problem is converted to standard LP form:

$$\text{minimize } f = -x_1 - 0.5x_2 \quad (\text{a})$$

subject to

$$2x_1 + 3x_2 + x_3 = 12 \quad (\text{b})$$

$$2x_1 + x_2 + x_4 = 8 \quad (\text{c})$$

$$x_i \geq 0; \quad i = 1 \text{ to } 4 \quad (\text{d})$$

Table 6-12 contains iterations of the Simplex method. The optimum point is reached in just one iteration because all the reduced cost coefficients are nonnegative in the second canonical form (second tableau). The solution is given as

$$\text{basic variables:} \quad x_1 = 4, x_3 = 4$$

$$\text{nonbasic variables:} \quad x_2 = x_4 = 0$$

$$\text{optimum cost function:} \quad f = -4$$

The solution corresponds to point B in Fig. 3-7. In the second tableau, the *reduced cost coefficient for the nonbasic variable x_2 is zero*. This means that it is possible to make x_2 basic without any change in the optimum cost function value. This suggests existence of *multiple optimum solutions*. Performing the pivot operation in column 2, we find another solution given in the third tableau of Table 6-12 as:

$$\text{basic variables:} \quad x_1 = 3, x_2 = 2$$

$$\text{nonbasic variables:} \quad x_3 = x_4 = 0$$

$$\text{optimum cost function:} \quad f = -4$$

TABLE 6-12 Solution by the Simplex Method for Example 6.9

Initial tableau: x_4 is identified to be replaced with x_1 in the basic set.

Basic↓	x_1	x_2	x_3	x_4	b	Ratio: b_i/a_{iq}
x_3	2	3	1	0	12	$\frac{12}{2} = 6$
x_4	2	1	0	1	8	$\frac{8}{2} = 4 \leftarrow$ smallest
Cost	-1	-0.5	0	0	$f - 0$	

Second tableau: First optimum point; reduced cost coefficients in nonbasic columns are nonnegative; the tableau gives optimum solution. $c_3 = 0$ indicates the possibility of multiple solutions. x_3 is identified to be replaced with x_2 in the basic set to obtain another optimum point.

Basic↓	x_1	x_2	x_3	x_4	b	Ratio: b_i/a_{iq}
x_3	0	2	1	-1	4	$\frac{4}{2} = 2 \leftarrow$ smallest
x_1	1	$\frac{1}{2}$	0	$\frac{1}{2}$	4	$\frac{4}{1/2} = 8$
Cost	0	0	0	$\frac{1}{2}$	$f + 4$	

Third tableau: Second optimum point.

Basic↓	x_1	x_2	x_3	x_4	b	Ratio: b_i/a_{iq}
x_2	0	1	$\frac{1}{2}$	$-\frac{1}{2}$	2	Not needed
x_1	1	0	$-\frac{1}{4}$	$\frac{3}{4}$	3	Not needed
Cost	0	0	0	$\frac{1}{2}$	$f + 4$	

This solution corresponds to point C on Fig. 3-7. Note that any point on the line B-C also gives an optimum solution. Multiple solutions can occur when the cost function is parallel to one of the constraints. For the present example, the cost function is parallel to the second constraint, which is active at the solution.

In general, if a reduced cost coefficient corresponding to a nonbasic variable is zero in the final tableau, there is a possibility of multiple optimum solutions. From a practical standpoint, this is not a bad situation. Actually, it may be desirable because it gives the designer options; any suitable point on the straight line joining the two optimum designs can be selected to better suit the needs of the designer. Note that all optimum design points are global solutions as opposed to local solutions.

Example 6.10 demonstrates how to recognize an unbounded feasible set (solution) for a problem.

EXAMPLE 6.10 Identification of an Unbounded Problem with the Simplex Method

Solve the LP problem: maximize $z = x_1 - 2x_2$ subject to $2x_1 - x_2 \geq 0$, $-2x_1 + 3x_2 \leq 6$, $x_1, x_2 \geq 0$

Solution. The problem has been solved graphically in Section 3.5. It can be seen from the graphical solution (Fig. 3-8) that the problem is unbounded. We will solve the problem using the Simplex method and see how we can recognize unbounded problems. Writing the problem in the standard Simplex form, we obtain the initial canonical form shown in Table 6-13 where x_3 and x_4 are the slack variables (note that the first constraint has been transformed as $-2x_1 + x_2 \leq 0$). The basic feasible solution is

$$\begin{aligned} \text{basic variables:} & \quad x_3 = 0, x_4 = 5 \\ \text{nonbasic variables:} & \quad x_1 = x_2 = 0 \\ \text{cost function:} & \quad f = 0 \end{aligned}$$

TABLE 6-13 Initial Canonical form for Example 6.10 (Unbounded Problem)

Basic↓	x_1	x_2	x_3	x_4	b
x_3	-2	1	1	0	0
x_4	-2	3	0	1	6
Cost	-1	2	0	0	$f - 0$

Scanning the last row, we find that the reduced cost coefficient for the nonbasic variable x_1 is negative. Therefore, x_1 should become a basic variable. However, a pivot element cannot be selected in the first column because there is no positive element. There is no other possibility of selecting another nonbasic variable to become basic; the reduced cost coefficient for x_2 (the other nonbasic variable) is positive. Therefore, no pivot steps can be performed, and yet we are not at the optimum point. Thus, the feasible set for the problem is unbounded. The foregoing observation will be true in general. For *unbounded problems*, there will be negative reduced cost coefficients for nonbasic variables but no possibility of pivot steps.

6.4 Two-Phase Simplex Method—Artificial Variables

The basic Simplex method of Section 6.3 is extended to handle “ \geq type” and equality constraints. A basic feasible solution is needed to initiate the Simplex solution process. Such a solution is immediately available if only “ \leq type” constraints are present. However, for the “ \geq type” and equality constraints, an initial basic feasible solution is not available. To obtain such a solution, we must introduce artificial variables for the “ \geq type” and equality constraints, define an auxiliary minimization LP problem, and solve it. The standard Simplex method can still be used to solve the auxiliary problem. This is called Phase I of the Simplex

procedure. At the end of Phase I, a basic feasible solution for the original problem becomes available. Phase II then continues to find a solution to the original LP problem. The method is illustrated with examples.

6.4.1 Artificial Variables

When there are “ \geq type” constraints in the LP problem, surplus variables are subtracted from them to transform the problem to the standard form. The equality constraints, if present, are not changed because they are already in the standard form. For such problems, an initial basic solution cannot be obtained by selecting the original design variables as nonbasic (setting them to zero), as is the case when there are only “ \leq type” constraints, e.g., for all the examples in Section 6.3. To obtain an initial basic feasible solution, the Gauss-Jordan elimination procedure can be used to convert the $\mathbf{Ax} = \mathbf{b}$ in the canonical form. However, an easier way is to introduce nonnegative auxiliary variables for the “ \geq type” and equality constraints, define an auxiliary LP problem, and solve it using the Simplex method. The auxiliary variables are called *artificial variables* and are different from the surplus variables. They have no physical meaning; however, with their addition we obtain an initial basic feasible solution for the auxiliary LP problem by treating them as basic along with any slack variables for “ \leq type” constraints. All other variables are treated as nonbasic (i.e., set to zero).

For the sake of simplicity of discussion, let us assume that each constraint of the standard LP problem requires an artificial variable. We shall see later in examples that constraints that do not require an artificial variable can also be treated routinely. Recalling that the standard LP problem has n variables and m constraints, the constraint equations augmented with the artificial variables are written as

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n + x_{n+1} &= b_1 \\ \cdot & \quad \cdot \quad \dots \quad \cdot \quad \cdot \\ \cdot & \quad \cdot \quad \dots \quad \cdot \quad \cdot \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n + x_{n+m} &= b_m \end{aligned} \tag{6.18}$$

where x_{n+j} , $j = 1$ to m are the artificial variables. The constraints of Eq. (6.18) can be written in the summation notation as

$$\sum_{j=1}^n a_{ij}x_j + x_{n+i} = b_i; \quad i = 1 \text{ to } m \tag{6.19}$$

Thus the initial basic feasible solution is obtained as $x_j = 0$, $j = 1$ to n , and $x_{n+i} = b_i$, $i = 1$ to m . Note that the artificial variables basically augment the convex polyhedron of the original problem. The initial basic feasible solution corresponds to an extreme point (vertex) located in the new expanded space. The problem now is to traverse extreme points in the expanded space until an extreme point is reached in the original space. When the original space is reached, all artificial variables will be nonbasic (i.e., they will have zero value). At this point the augmented space is literally removed so that future movements are only among the extreme points of the original space until the optimum is reached. In short, after creating artificial variables, we eliminate them as quickly as possible. The preceding procedure is called the *two-phase Simplex method* of LP.

6.4.2 Artificial Cost Function

To eliminate the artificial variables from the problem, we define an auxiliary function called the *artificial cost function*, and minimize it subject to the constraints of Eq. (6.19) and non-negativity of all the variables. The artificial cost function is simply a sum of all the artificial variables and will be designated as w :

$$w = x_{n+1} + x_{n+2} + \dots + x_{n+m} = \sum_{i=1}^m x_{n+i} \quad (6.20)$$

6.4.3 Definition of Phase I Problem

Since the artificial variables are introduced to simply obtain an initial basic feasible solution for the original problem, they need to be eliminated eventually. This elimination is done by defining an LP problem called the Phase I problem. The objective of this problem is to make all the artificial variables nonbasic so that they have zero value. In that case, the artificial cost function in Eq. (6.20) will be zero, indicating the end of Phase I. However the Phase I problem is not yet in a form suitable to initiate the Simplex method. The reason is that the reduced cost coefficients c'_j of the nonbasic variables in the artificial cost function are not yet available to determine the pivot element and perform the pivot step. Currently, the artificial cost function in Eq. (6.20) is in terms of the basic variables x_{n+1}, \dots, x_{n+m} . Therefore the reduced cost coefficients c'_j cannot be identified. They can be identified only if the artificial cost function w is in terms of the nonbasic variables x_1, \dots, x_n . To obtain w in terms of nonbasic variables, we use the constraint expressions to eliminate the basic variables from the artificial cost function. Calculating x_{n+1}, \dots, x_{n+m} from Eqs. (6.18) and substituting into Eq. (6.20), we obtain the artificial cost function w in terms of the nonbasic variables as

$$w = \sum_{i=1}^m b_i - \sum_{j=1}^n \sum_{i=1}^m a_{ij} x_j \quad (6.21)$$

The reduced cost coefficients c'_j are identified as the coefficients of the nonbasic variables x_j in Eq. (6.21) as

$$c'_j = -\sum_{i=1}^m a_{ij}; \quad j = 1 \text{ to } n \quad (6.22)$$

If there are also “ \leq type” constraints in the original problem, these are cast into the standard LP form by adding slack variables that serve as basic variables in Phase I. Therefore, the number of artificial variables is less than m —the total number of constraints. Accordingly, the number of artificial variables required to obtain an initial basic feasible solution is also less than m . This implies that the sums in Eqs. (6.21) and (6.22) are not for all the m constraints. They are only over the constraints requiring an artificial variable.

6.4.4 Phase I Algorithm

The standard Simplex procedure described in Section 6.3 can now be employed to solve the auxiliary optimization problem of Phase I. During this phase, the artificial cost function is used to determine the pivot element. The original cost function is treated as a constraint and the elimination step is also executed for it. This way, the real cost function is in terms of the nonbasic variables only at the end of Phase I, and the Simplex method can be continued during Phase II. All artificial variables become nonbasic at the end of Phase I. Since w is the sum of all the artificial variables, its minimum value is clearly zero. When $w = 0$, an extreme point of the original feasible set is reached. w is then discarded in favor of f and iterations continue in Phase II until the minimum of f is obtained. Suppose, however, that w cannot be driven to zero. This will be apparent when none of the reduced cost coefficients for the artificial cost function is negative and yet w is greater than zero. Clearly, this means that we cannot reach the original feasible set and, therefore, *no feasible solution exists for the original design problem, i.e., it is an infeasible problem*. At this point the designer should re-examine the formulation of the problem, which may be over-constrained or improperly formulated.

6.4.5 Phase II Algorithm

In the final tableau from Phase I, the artificial cost row is replaced by the actual cost function equation and the Simplex iterations continue based on the algorithm explained in Section 6.3. The basic variables, however, should not appear in the cost function. Thus, pivot steps need to be performed on the cost function equation to eliminate the basic variables from it. A convenient way of accomplishing this is to treat the cost function as one of the equations in the Phase I tableau, say the second equation from the bottom. Elimination is performed on this equation along with others. In this way, the cost function is in the correct form to continue with Phase II. The artificial variable columns can also be discarded for Phase II calculations. However, they are kept in the tableau because they provide information that can be useful for postoptimality analysis.

For most LP problems, the Simplex method yields one of the following results as illustrated in the examples:

1. If there is a solution to the problem, the method will find it (Example 6.11).
2. If the problem is infeasible, the method will indicate that (Example 6.12).
3. If the problem is unbounded, the method will indicate that (Example 6.10, Example 6.13).
4. If there are multiple solutions, the method will indicate that, as seen in Examples 6.6 and 6.9.

EXAMPLE 6.11 Use of Artificial Variable for “ \geq Type” Constraints

Find the optimum solution for the following LP problem using the Simplex method: maximize $z = y_1 + 2y_2$ subject to $3y_1 + 2y_2 \leq 12$, $2y_1 + 3y_2 \geq 6$, $y_1 \geq 0$, y_2 is unrestricted in sign.

Solution. The graphical solution for the problem is shown in Fig. 6-4. It can be seen that the optimum solution is at point B. We shall use the two-phase Simplex method to verify the solution. Since y_2 is free in sign, we decompose it as $y_2 = y_2^+ - y_2^-$. To write the problem in the standard form, we define $x_1 = y_1$, $x_2 = y_2^+$, and $x_3 = y_2^-$, and transform the problem as

$$\text{minimize } f = -x_1 - 2x_2 + 2x_3 \quad (\text{a})$$

subject to

$$3x_1 + 2x_2 - 2x_3 + x_4 = 12 \quad (\text{b})$$

$$2x_1 + 3x_2 - 3x_3 - x_5 = 6 \quad (\text{c})$$

$$x_i \geq 0; \quad i = 1 \text{ to } 5 \quad (\text{d})$$

where x_4 is a slack variable for the first constraint and x_5 is a surplus variable for the second constraint. It can be seen that if we select the real variables as nonbasic, i.e., $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, the resulting basic solution is infeasible because $x_5 = -6$. Therefore, we need to use the two-phase algorithm. Accordingly, we introduce an artificial variable x_6 in the second constraint as

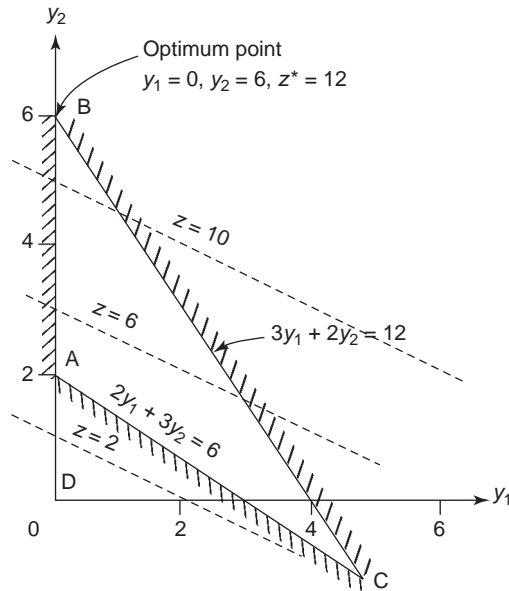


FIGURE 6-4 Graphical solution for Example 6.11.

$$2x_1 + 3x_2 - 3x_3 - x_5 + x_6 = 6 \quad (e)$$

The artificial cost function is defined as $w = x_6$. Since w should be in terms of nonbasic variables (x_6 is basic), we substitute for x_6 from Eq. (e) and obtain w as

$$w = x_6 = 6 - 2x_1 - 3x_2 + 3x_3 + x_5 \quad (f)$$

The initial tableau for Phase I is shown in Table 6-14. The initial basic variables are $x_4 = 12$ and $x_6 = 6$. The nonbasic variables are $x_1 = x_2 = x_3 = x_5 = 0$. Also $w = 6$ and $f = 0$. This corresponds to the infeasible point D in Fig. 6-4. According to the Simplex algorithm, the pivot element is a_{22} , which implies that x_2 should become basic and x_6 should become nonbasic. Performing the pivot step, we obtain the second tableau given in Table 6-14. For the second tableau, $x_4 = 8$ and $x_2 = 2$ are the basic variables and all others are nonbasic. This corresponds to the feasible point A in Fig. 6-4. Since all the reduced cost coefficients of the artificial cost function are nonnegative and the artificial cost function is zero, an initial basic feasible solution for the original problem is obtained. Therefore, this is the end of Phase I.

For Phase II, column x_6 should be ignored in determining pivots. For the next step, the pivot element is a_{15} in the second tableau according to Steps 1 and 2 of the Simplex method. This implies that x_4 should be replaced by x_5 as a basic variable. The third tableau is obtained as shown in Table 6-14. The last tableau yields an optimum solution for the problem, which is $x_5 = 12$, $x_2 = 6$, $x_1 = x_3 = x_4 = 0$, and $f = -12$. The solution for the original design problem is then $y_1 = 0$, $y_2 = 6$, and $z = 12$, which agrees with the graphical solution of Fig. 6-4. Note that the artificial variable column (x_6) in the final tableau is the negative of the surplus variable column (x_5). This is true for all “ \geq type” constraints.

TABLE 6-14 Solution by the Two-Phase Simplex Method for Example 6.11

Initial tableau: x_6 is identified to be replaced with x_2 in the basic set.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	b	Ratio
x_4	3	2	-2	1	0	0	12	$\frac{12}{2} = 6$
x_6	2	3	-3	0	-1	1	6	$\frac{6}{3} = 2$
Cost	-1	-2	2	0	0	0	$f - 0$	
Artificial cost	-2	-3	3	0	1	0	$w - 6$	

Second tableau: End of Phase I. Begin Phase II. x_4 is identified to be replaced with x_5 in the basic set.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	b	Ratio
x_4	$\frac{5}{3}$	0	0	1	$\frac{2}{3}$	$-\frac{2}{3}$	8	$\frac{8}{2/3} = 12$
x_2	$\frac{2}{3}$	1	-1	0	$-\frac{1}{3}$	$\frac{1}{3}$	2	Negative
Cost	$\frac{1}{3}$	0	0	0	$-\frac{2}{3}$	$\frac{2}{3}$	$f + 4$	
Artificial cost	0	0	0	0	0	1	$w - 0$	

Third tableau: Reduced cost coefficients in nonbasic columns are nonnegative; the third tableau gives the optimum solution. End of Phase II.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	b
x_5	$\frac{5}{2}$	0	0	$\frac{3}{2}$	1	-1	12
x_2	$\frac{3}{2}$	1	-1	$\frac{1}{2}$	0	0	6
Cost	2	0	0	1	0	0	$f + 12$

EXAMPLE 6.12 Use of Artificial Variables for Equality Constraints (Infeasible Problem)

Solve the LP problem: maximize $z = x_1 + 4x_2$ subject to $x_1 + 2x_2 \leq 5$, $2x_1 + x_2 = 4$, $x_1 - x_2 \geq 3$, $x_1, x_2 \geq 0$.

Solution. The constraints for the problem are plotted in Fig. 6-5. It can be seen that the problem has no feasible solution. We will solve the problem using the Simplex method to see how we can recognize an infeasible problem. Writing the problem in the standard LP form, we obtain

$$\text{minimize } f = -x_1 - 4x_2 \quad (\text{a})$$

subject to

$$x_1 + 2x_2 + x_3 = 5 \quad (\text{b})$$

$$2x_1 + x_2 + x_5 = 4 \quad (\text{c})$$

$$x_1 - x_2 - x_4 + x_6 = 3 \quad (\text{d})$$

$$x_i \geq 0; \quad i = 1 \text{ to } 6 \quad (\text{e})$$

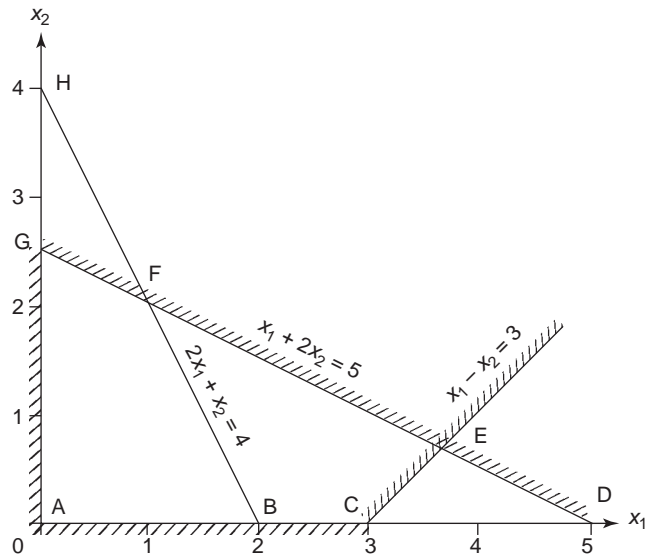


FIGURE 6-5 Constraints for Example 6.12. Infeasible problem.

TABLE 6-15 Solution for Example 6.12 (Infeasible Problem)

Initial tableau: x_5 is identified to be replaced with x_1 in the basic set.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	b	Ratio
x_3	1	2	1	0	0	0	5	$\frac{5}{1} = 5$
x_5	2	1	0	0	1	0	4	$\frac{4}{2} = 2$
x_6	1	-1	0	-1	0	1	3	$\frac{3}{1} = 3$
Cost	-1	-4	0	0	0	0	$f - 0$	
Artificial cost	-3	0	0	1	0	0	$w - 7$	

Second tableau: End of Phase I.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	b
x_3	0	$\frac{3}{2}$	1	0	$-\frac{1}{2}$	0	3
x_1	1	$\frac{1}{2}$	0	0	$\frac{1}{2}$	0	2
x_6	0	$-\frac{3}{2}$	0	-1	$-\frac{1}{2}$	1	1
Cost	0	$-\frac{7}{2}$	0	0	$\frac{1}{2}$	0	$f + 2$
Artificial cost	0	$\frac{3}{2}$	0	1	$\frac{3}{2}$	0	$w - 1$

Here x_3 is a slack variable, x_4 is a surplus variable, and x_5 and x_6 are artificial variables. Table 6-15 shows Phase I iterations of the Simplex method. It can be seen that after the first pivot step, all the reduced cost coefficients of the artificial cost function for nonbasic variables are positive indicating the end of Phase I. However, the artificial cost function is not zero ($w = 1$). Therefore there is no feasible solution to the original problem.

EXAMPLE 6.13 Use of Artificial Variables (Unbounded Problem)

Solve the LP problem: maximize $z = 3x_1 - 2x_2$ subject to $x_1 - x_2 \geq 0$, $x_1 + x_2 \geq 2$, $x_1, x_2 \geq 0$.

Solution. The constraints for the problem are plotted in Fig. 6-6. It can be seen that the problem is unbounded. We will solve the problem by the Simplex method and see how to recognize *unboundedness*. Transforming the problem to the standard form, we get:

$$\text{minimize } f = -3x_1 + 2x_2 \quad (\text{a})$$

subject to

$$-x_1 + x_2 + x_3 = 0 \quad (\text{b})$$

$$x_1 + x_2 - x_4 + x_5 = 2 \quad (\text{c})$$

$$x_i \geq 0; \quad i = 1 \text{ to } 5 \quad (\text{d})$$

where x_3 is a slack variable, x_4 is a surplus variable, and x_5 is an artificial variable. Note that the right side of the first constraint is zero so it can be treated as either “ \leq type” or “ \geq type.” We will treat it as “ \leq type.” Note also that the second constraint is “ \geq type,” so we must use an artificial variable and an artificial cost function to find the initial basic feasible solution. The solution for the problem is given in Table 6-16. For the initial tableau $x_3 = 0$ and $x_5 = 2$ are basic variables and all others are nonbasic. Note that this is a *degenerate basic feasible solution*. The solution corresponds to point A (the origin) in Fig. 6-6. Scanning the artificial cost row, we observe that there are two possibilities for pivot columns, x_1 or x_2 . If x_2 is selected as the pivot column,

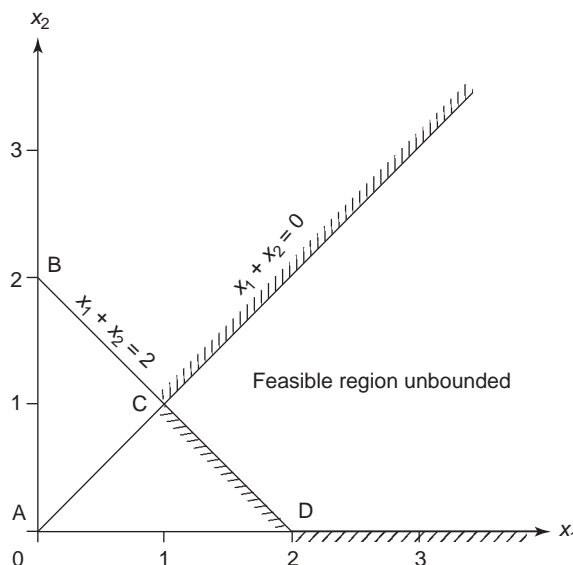


FIGURE 6-6 Constraints for Example 6.13. Unbounded problem.

TABLE 6-16 Solution for Example 6.13 (Unbounded Problem)

Initial tableau: x_5 is identified to be replaced with x_1 in the basic set.

Basic↓	x_1	x_2	x_3	x_4	x_5	b	Ratio
x_3	-1	1	1	0	0	0	Negative
x_5	1	1	0	-1	1	2	$\frac{2}{1} = 2$
Cost	-3	2	0	0	0	$f - 0$	
Artificial cost	-1	-1	0	1	0	$w - 2$	

Second tableau: End of Phase I. End of Phase II.

Basic↓	x_1	x_2	x_3	x_4	x_5	b	Ratio
x_3	0	2	1	-1	1	2	Negative
x_1	1	1	0	-1	1	2	Negative
Cost	0	5	0	-3	3	$f + 6$	
Artificial cost	0	0	0	0	1	$w - 0$	

then the first row must be the pivot row with $a_{12} = 1$ as the pivot element. This will make x_2 basic and x_3 nonbasic. However, x_2 will remain zero and the resulting solution will be degenerate, corresponding to point A. One more iteration will be necessary to move from A to D. If we choose x_1 as the pivot column, then $a_{21} = 1$ will be the pivot element making x_1 as basic and x_5 as nonbasic. Carrying out the pivot step, we obtain the second tableau as shown in Table 6-16. The basic feasible solution is $x_1 = 2$, $x_3 = 2$, and other variables are zero. This solution corresponds to point D in Fig. 6-6. This is the basic feasible solution for the original problem because the artificial cost function is zero, i.e., $w = 0$. The original cost function has also reduced from 0 to -6 . This is the end of Phase I. Scanning the cost function row, we find that the reduced cost coefficient c'_4 is negative, but the pivot element cannot be determined, i.e., x_4 cannot be made basic (all elements in the x_4 column are negative in the second tableau). This indicates the problem to be unbounded.

6.4.6 Degenerate Basic Feasible Solution

It is possible that during iterations of the Simplex method, a basic variable attains zero value, i.e., the basic feasible solution becomes degenerate. What are the implications of this situation? We shall discuss them in Example 6.14.

EXAMPLE 6.14 Implications of Degenerate Basic Feasible Solution

Solve the following LP problem by the Simplex method: maximize $z = x_1 + 4x_2$ subject to $x_1 + 2x_2 \leq 5$, $2x_1 + x_2 \leq 4$, $2x_1 + x_2 \geq 4$, $x_1 - x_2 \geq 1$, $x_1, x_2 \geq 0$.

Solution. The problem is transcribed into the standard LP form as follows:

$$\text{minimize } f = -x_1 - 4x_2 \quad (\text{a})$$

subject to

$$x_1 + 2x_2 + x_3 = 5 \quad (\text{b})$$

$$2x_1 + x_2 + x_4 = 4 \quad (\text{c})$$

$$2x_1 + x_2 - x_5 + x_7 = 4 \quad (\text{d})$$

$$x_1 - x_2 - x_6 + x_8 = 1 \quad (\text{e})$$

$$x_i \geq 0; \quad i = 1 \text{ to } 8 \quad (\text{f})$$

where x_3 and x_4 are slack variables, x_5 and x_6 are surplus variables, and x_7 and x_8 are artificial variables. The two-phase Simplex procedure takes three iterations to reach the optimum point. These iterations are given in Table 6-17. It can be seen that in the third tableau, the basic variable x_4 has zero value so the *basic feasible solution is degenerate*. At this iteration, it is determined that x_5 should become basic so x_5 is the pivot column. We need to determine the pivot row. We take ratios of the right sides with the positive elements in the x_5 column. This determines the second row as the pivot row because it has the lowest ratio (zero). In general, if the element in the pivot column and the row that gives degenerate basic variable is positive, then that row must always be the pivot row; otherwise, the new solution cannot be feasible. Also, in this case, the new basic feasible solution will be degenerate, as for the final tableau in Table 6-17. The only way the new feasible solution can be nondegenerate is when the element in the pivot column and the degenerate variable row is negative. In that case

TABLE 6-17 Solution for Example 6.14 (Degenerate Basic Feasible Solution)

Initial tableau: x_8 is identified to be replaced with x_1 in the basic set.										
Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	b	Ratio
x_3	1	2	1	0	0	0	0	0	5	$\frac{5}{1} = 5$
x_4	2	1	0	1	0	0	0	0	4	$\frac{4}{2} = 2$
x_7	2	1	0	0	-1	0	1	0	4	$\frac{4}{2} = 2$
x_8	1	-1	0	0	0	-1	0	1	1	$\frac{1}{1} = 1$
Cost	-1	-4	0	0	0	0	0	0	$f - 0$	
Artificial	-3	0	0	0	1	1	0	0	$w - 5$	
Second tableau: x_7 is identified to be replaced with x_2 in the basic set.										
Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	b	Ratio
x_3	0	3	1	0	0	1	0	-1	4	$\frac{4}{3}$
x_4	0	3	0	1	0	2	0	-2	2	$\frac{2}{3}$
x_7	0	3	0	0	-1	2	1	-2	2	$\frac{2}{3}$
x_1	1	-1	0	0	0	-1	0	1	1	Negative
Cost	0	-5	0	0	0	-1	0	1	$f + 1$	
Artificial	0	-3	0	0	1	-2	0	3	$w - 2$	

TABLE 6-17 Continued

Third tableau: x_4 is identified to be replaced with x_5 in the basic set. End of Phase I.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	b	Ratio
x_3	0	0	1	0	1	-1	-1	1	2	$\frac{2}{1} = 2$
x_4	0	0	0	1	1	0	-1	0	0	$\frac{0}{1} = 0$
x_2	0	1	0	0	$-\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{3}$	$-\frac{2}{3}$	$\frac{2}{3}$	Negative
x_1	1	0	0	0	$-\frac{1}{3}$	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{5}{3}$	Negative
Cost	0	0	0	0	$-\frac{5}{3}$	$\frac{7}{3}$	$\frac{5}{3}$	$-\frac{7}{3}$	$f + \frac{13}{3}$	
Artificial	0	0	0	0	0	0	1	1	$w - 0$	

Final tableau: End of Phase II.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	b
x_3	0	0	1	-1	0	-1	0	1	2
x_5	0	0	0	1	1	0	-1	0	0
x_2	0	1	0	$\frac{1}{3}$	0	$\frac{2}{3}$	0	$-\frac{2}{3}$	$\frac{2}{3}$
x_1	1	0	0	$\frac{1}{3}$	0	$-\frac{1}{3}$	0	$\frac{1}{3}$	$\frac{5}{3}$
Cost	0	0	0	$\frac{5}{3}$	0	$\frac{7}{3}$	0	$-\frac{7}{3}$	$f + \frac{13}{3}$

the new basic feasible solution will be nondegenerate. *It is theoretically possible for the Simplex method to fail by cycling between two degenerate basic feasible solutions.* However, in practice this usually does not happen. The final solution for this problem is

basic variables: $x_1 = \frac{5}{3}, x_2 = \frac{2}{3}, x_3 = 2, x_5 = 0$

nonbasic variables: $x_4 = x_6 = x_7 = x_8 = 0$

optimum cost function: $f = -\frac{3}{13}$ or $z = \frac{13}{3}$

6.5 Postoptimality Analysis

The optimum solution of the LP problem depends on the parameters in vectors **c** and **b**, and the matrix **A** defined in Eqs. (6.9) to (6.11). These parameters are prone to errors in practical design problems. Thus we are interested not only in the optimum solution but also in how it changes when the parameters change. The changes may be either discrete (e.g., when we are uncertain about which of several choices is the value of a particular parameter) or continuous. *The study of discrete parameter changes is often called sensitivity analysis, and that of continuous changes is called parametric programming.* There are five basic parametric changes affecting the solution:

1. Changes in the cost function coefficients, c_j
2. Changes in the resource limits, b_i
3. Changes in the constraint coefficients, a_{ij}
4. The effect of including additional constraints
5. The effect of including additional variables

A thorough discussion of these changes, while not necessarily difficult, is beyond our scope. In principle, we could imagine solving a new problem for every change. Fortunately, for a small number of changes there are useful shortcuts. Almost all computer programs for LP problems provide some information about parameter variations. We shall study the parametric changes defined in items 1 through 3. *The final tableau for the LP problem contains all the information needed to study these changes.* We shall describe the information contained in the final tableau and its use to study the three parametric changes. For other variations, full length texts on linear programming may be consulted.

It turns out that the optimum solution of the altered problem can be computed using the optimum solution of the original problem and the information in the final tableau as long as changes in the parameters are within certain limits. This is especially beneficial for problems that take a long time to solve. In the following discussion we use a'_{ij} , c'_j , and b'_i to represent the corresponding values of the parameters a_{ij} , c_j , and b_i in the final tableau.

6.5.1 Changes in Resource Limits

First, we study how the optimum value of the cost function for the problem changes if we change the right side parameters, b_i 's (also known as *resource limits*), of the constraints. The constraint variation sensitivity Theorem 4.7 of Section 4.5 can be used to study the effect of these changes. Use of that theorem requires knowledge of the Lagrange multipliers for the constraints. Theorem 6.5 gives a way of recovering the multipliers for the constraints of an LP problem from the *final tableau*.

Theorem 6.5 Lagrange Multiplier Values Let the standard LP problem be solved using the Simplex method. (1) For “ \leq type” constraints, the Lagrange multiplier equals the reduced cost coefficient in the slack variable column associated with the constraint. (2) For “ $=$ ” and “ \geq type” constraints, the Lagrange multiplier equals the reduced cost coefficient in the artificial variable column associated with the constraint. (3) The Lagrange multiplier is always ≥ 0 for the “ \leq type” constraint, always ≤ 0 for the “ \geq type” constraint, and free in sign for the “ $=$ type” constraint.

In Section 4.5, the *physical meaning of the Lagrange multipliers* was described. There, the Lagrange multipliers were related to derivatives of the cost function with respect to the right side parameters. Equality and inequality constraints were treated separately with v_i and u_i as their Lagrange multipliers, respectively. However, in this section, we use a slightly different notation. We use e_i as the right side parameter of any constraint and y_i as its Lagrange multiplier. Using this notation and Theorem 4.7, we obtain the following derivative of the cost function with respect to the right side parameters, and change in the optimum cost function:

$$\frac{\partial f}{\partial e_i} = -y_i; \quad \Delta f = -y_i \Delta e_i = -y_i (e_{\text{new}} - e_{\text{old}}) \quad (6.23)$$

It is noted here that Theorem 6.5 and Eq. (6.23) are *applicable only if changes in the right side parameters are within certain limits*, i.e., there are upper and lower limits on changes in the resource limits for which Eq. (6.23) is valid. The changes need not be small any more as was stipulated for nonlinear problems in Section 4.5. Calculations for the limits are dis-

cussed later in this section. Note the calculation for Δf remains valid for simultaneous changes to multiple constraints; in that case all the changes are added.

It is also noted that *Theorem 4.7 and Eq. (6.23) were discussed for the general problem written as minimization of a cost function with “ \leq type” and equality constraints.* However, Eq. (6.23) is applicable to “ \geq type” constraints as well as long as care is exercised in using appropriate signs for the Lagrange multiplier y_i and the change Δe_i . We shall demonstrate use of Theorem 6.5 and Eq. (6.23) with examples.

It is also important to note that if an inequality is inactive at the optimum, then its slack or surplus variable is greater than 0. Therefore its *Lagrange multiplier is 0 to satisfy the switching condition*, $y_i s_i = 0$ (except for the abnormal case where both the Lagrange multiplier and the constraint function have zero value). This observation can help in verifying the correctness of the Lagrange multipliers recovered from the final LP tableau. Example 6.15 describes recovery of the Lagrange multipliers from the final tableau for the “ \leq type” constraints.

EXAMPLE 6.15 Recovery of Lagrange Multipliers for “ \leq Type” Constraint

Consider the problem: maximize $z = 5x_1 - 2x_2$ subject to the constraints $2x_1 + x_2 \leq 9$, $x_1 - 2x_2 \leq 2$, $-3x_1 + 2x_2 \leq 3$, $x_1, x_2 \geq 0$. Solve the problem by the Simplex method. Recover Lagrange multipliers for the constraints.

Solution. Constraints for the problem and cost function contours are plotted in Fig. 6-7. The optimum solution is at point C and is given as $x_1 = 4$, $x_2 = 1$, $z = 18$. The problem is transformed to the standard form as

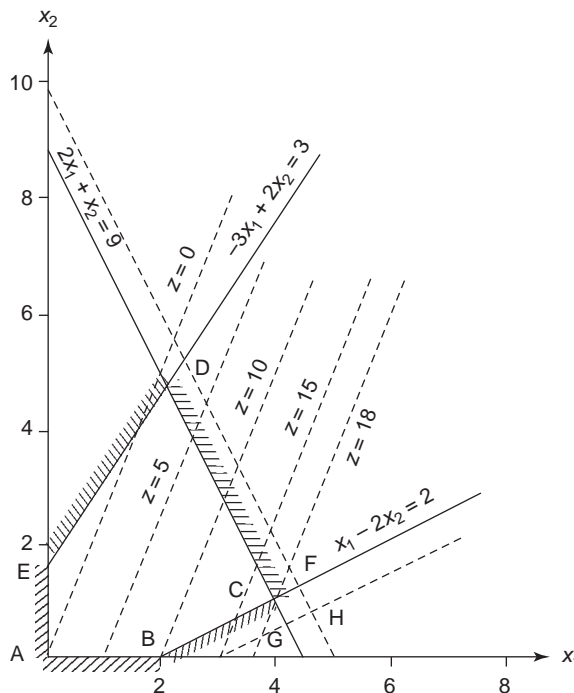


FIGURE 6-7 Graphical solution for Example 6.15.

$$\text{minimize } f = -5x_1 + 2x_2 \quad (\text{a})$$

subject to

$$2x_1 + x_2 + x_3 = 9 \quad (\text{b})$$

$$x_1 - 2x_2 + x_4 = 2 \quad (\text{c})$$

$$-3x_1 + 2x_2 + x_5 = 3 \quad (\text{d})$$

$$x_i \geq 0; \quad i = 1 \text{ to } 5 \quad (\text{e})$$

where x_3 , x_4 , and x_5 are the slack variables. Solving the problem using the Simplex method, we obtain the sequence of calculations given in Table 6-18. From the final tableau,

$$\text{basic variables: } \quad x_1 = 4, \quad x_2 = 1, \quad x_5 = 13$$

$$\text{nonbasic variables: } \quad x_3 = 0, \quad x_4 = 0$$

$$\text{objective function: } \quad z = 18 \quad (f = -18)$$

In the problem formulation, x_3 , x_4 , and x_5 are the slack variables for the three constraints. Since all constraints are “ \leq type,” the reduced cost coefficients for the slack variables are the Lagrange multipliers as follows:

TABLE 6-18 Solution for Example 6.15 by the Simplex Method

Initial tableau: x_4 is identified to be replaced with x_1 in the basic set.

Basic↓	x_1	x_2	x_3	x_4	x_5	b
x_3	2	1	1	0	0	9
x_4	1	-2	0	1	0	2
x_5	-3	2	0	0	1	3
Cost	-5	2	0	0	0	$f - 0$

Second tableau: x_3 is identified to be replaced with x_2 in the basic set.

Basic↓	x_1	x_2	x_3	x_4	x_5	b
x_3	0	5	1	-2	0	5
x_1	1	-2	0	1	0	2
x_5	0	-4	0	3	1	9
Cost	0	-8	0	5	0	$f + 10$

Third tableau: Reduced cost coefficients in nonbasic columns are nonnegative; the tableau gives optimum point.

Basic↓	x_1	x_2	x_3	x_4	x_5	b
x_2	0	1	0.2	-0.4	0	1
x_1	1	0	0.4	0.2	0	4
x_5	0	0	0.8	1.4	1	13
Cost	0	0	1.6	1.8	0	$f + 18$
	(c'_1)	(c'_2)	(c'_3)	(c'_4)	(c'_5)	

x_3 , x_4 , and x_5 are slack variables.

1. For $2x_1 + x_2 \leq 9$: $y_1 = 1.6$ (c'_3 in column x_3)
2. For $x_1 - 2x_2 \leq 2$: $y_2 = 1.8$ (c'_4 in column x_4)
3. For $-3x_1 + 2x_2 \leq 3$: $y_3 = 0$ (c'_5 in column x_5)

Therefore, Eq. (6.23) gives partial derivatives of f with respect to e_i as

$$\frac{\partial f}{\partial e_1} = -1.6; \quad \frac{\partial f}{\partial e_2} = -1.8; \quad \frac{\partial f}{\partial e_3} = 0 \quad (f)$$

where $f = -(5x_1 - 2x_2)$; note that Eq. (6.23) is valid for a minimization problem only. If the right side of the first constraint changes from 9 to 10, the cost function f changes by $\Delta f = -1.6(10 - 9) = -1.6$, i.e., the new value of f will be -19.6 ($z = 19.6$). Point F in Fig. 6-7 gives the new optimum solution for this case. If the right side of the second constraint changes from 2 to 3, the cost function f changes by $\Delta f = -1.8(3 - 2) = -1.8$ to -19.8 . Point G in Fig. 6-7 gives the new optimum solution. Note that any small change in the right side of the third constraint will have no effect on the cost function. *When the right side of first and second constraints are changed to 10 and 3 simultaneously*, the net change in the cost function is $-(1.6 + 1.8)$, i.e., new f will be -21.4 . The new solution is at point H in Fig. 6-7.

It is noted (as in Section 4.5) that the *Lagrange multipliers are very useful* for practical design problems. Their values give the relative effect of changes in the right side parameters of constraints (resource limits). Using their relative values, the designer can determine the most profitable way to adjust the resource limits, if necessary and possible. *The Lagrange multipliers are also called the dual variables (or, dual prices)*. The concept of duality in linear programming is described in Chapter 7. Example 6.16 demonstrates recovery of Lagrange multipliers for equality and “ \geq type” constraints.

EXAMPLE 6.16 Recovery of Lagrange Multipliers for “=” and “ \geq Type” Constraints

Solve the following LP problem and recover proper Lagrange multipliers for the constraints: maximize $z = x_1 + 4x_2$ subject to $x_1 + 2x_2 \leq 5$, $2x_1 + x_2 = 4$, $x_1 - x_2 \geq 1$, $x_1, x_2 \geq 0$

Solution. Constraints for the problem are plotted in Fig. 6-8. It can be seen that line E–C is the feasible region for the problem and point E gives the optimum solution. Converting the problem to standard Simplex form, we obtain:

$$\text{minimize } f = -x_1 - 4x_2 \quad (a)$$

subject to

$$x_1 + 2x_2 + x_3 = 5 \quad (b)$$

$$2x_1 + x_2 + x_5 = 4 \quad (c)$$

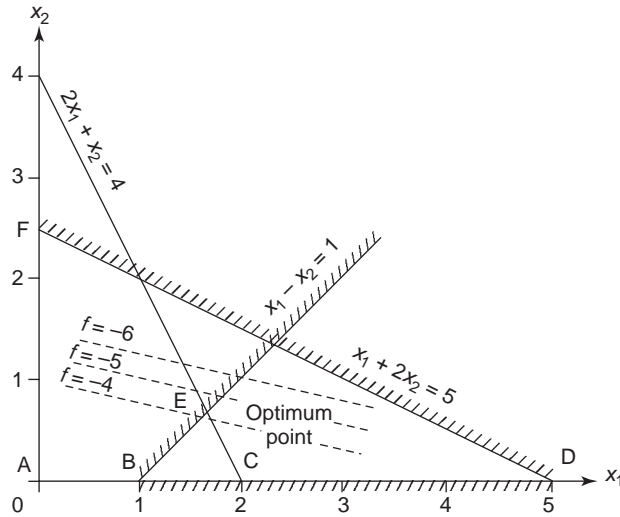


FIGURE 6-8 Constraints for Example 6.16. Feasible region: line E-C.

$$x_1 - x_2 - x_4 + x_6 = 1 \quad (d)$$

$$x_i \geq 0; \quad i = 1 \text{ to } 6 \quad (e)$$

where x_3 is a slack variable, x_4 is a surplus variable, and x_5 and x_6 are artificial variables. The problem, solved in Table 6-19, takes just two iterations to reach the optimum. The solution from the final tableau is

$$\begin{aligned} \text{basic variables:} & \quad x_1 = \frac{5}{3}, x_2 = \frac{2}{3}, x_3 = 2 \\ \text{nonbasic variables:} & \quad x_4 = 0, x_5 = 0, x_6 = 0 \\ \text{cost function:} & \quad f = -\frac{13}{3} \end{aligned}$$

Note that the artificial variable column (x_6) is the negative of the surplus variable column (x_4) for the third constraint. Using Theorem 6.5, the Lagrange multipliers for the constraints are

1. For $x_1 + 2x_2 \leq 5$: $y_1 = 0$ (c'_3 in the slack variable column x_3)
2. For $2x_1 + x_2 = 4$: $y_2 = \frac{5}{3}$ (c'_5 in the artificial variable column x_5)
3. For $x_1 - x_2 \geq 1$: $y_3 = -\frac{7}{3}$ (c'_6 in the artificial variable column x_6)

When the right side of the third constraint is changed to 2 (i.e., $x_1 - x_2 \geq 2$), the cost function $f = (-x_1 - 4x_2)$ changes by

$$\Delta f = -y_3 \Delta e_3 = -\left(-\frac{7}{3}\right)(2-1) = \frac{7}{3} \quad (f)$$

That is, the cost function will increase by $\frac{7}{3}$, from $-\frac{13}{3}$ to -2 ($z = 2$). This can be also observed from Fig. 6-8. We shall demonstrate that same result is obtained when the third constraint is written in the “ \leq form” ($-x_1 + x_2 \leq -1$). The Lagrange multiplier for the constraint is $\frac{7}{3}$, which is the negative of the preceding value. Note that it is also c'_4

TABLE 6-19 Solution for Example 6.16 with Equality Constraint

Initial tableau: x_6 is identified to be replaced with x_1 in the basic set.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	b
x_3	1	2	1	0	0	0	5
x_5	2	1	0	0	1	0	4
x_6	1	-1	0	-1	0	1	1
Cost	-1	-4	0	0	0	0	$f - 0$
Artificial	-3	0	0	1	0	0	$w - 5$

Second tableau: x_5 is identified to be replaced with x_2 in the basic set.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	b
x_3	0	3	1	1	0	-1	4
x_5	0	3	0	2	1	-2	2
x_1	1	-1	0	-1	0	1	1
Cost	0	-5	0	-1	0	1	$f + 1$
Artificial	0	-3	0	-2	0	3	$w - 2$

Third tableau: Reduced cost coefficients in nonbasic columns are nonnegative; the tableau gives the optimum point. End of Phase I. End of Phase II.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	b
x_3	0	0	1	-1	-1	1	2
x_2	0	1	0	$\frac{2}{3}$	$\frac{1}{3}$	$-\frac{2}{3}$	$\frac{2}{3}$
x_1	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{5}{3}$
Cost	0	0	0	$\frac{7}{3}$	$\frac{5}{3}$	$-\frac{7}{3}$	$f + \frac{13}{3}$
	(c'_1)	(c'_2)	(c'_3)	(c'_4)	(c'_5)	(c'_6)	
Artificial	0	0	0	0	1	1	$w - 0$

x_3 , slack variable; x_4 , surplus variable; x_5, x_6 , artificial variables.

in the surplus variable column x_4 . When the right side of the third constraint is changed to 2 (i.e., it becomes $-x_1 + x_2 \leq -2$), the cost function $f = (-x_1 - 4x_2)$ changes by

$$\Delta f = -y_3 \Delta e_3 = -\frac{7}{3}[-2 - (-1)] = \frac{7}{3} \quad (g)$$

which is same as before.

When the right side of the equality constraint is changed to 5 from 4, the cost function $f = (-x_1 - 4x_2)$ changes by

$$\Delta f = -y_2 \Delta e_2 = -\frac{5}{3}(5 - 4) = -\frac{5}{3} \quad (h)$$

That is, the cost function will decrease by $\frac{5}{3}$, from $-\frac{13}{3}$ to -6 ($z = 6$).

6.5.2 Ranging Right Side Parameters

When the right side of a constraint is changed, the constraint boundary moves parallel to itself, changing the feasible region for the problem. However, the isocost lines do not change. Since the feasible region is changed, the optimum solution may change, i.e., the design variables as well as the cost function may change. There are, however, certain limits on changes for which the set of active constraints at the optimum point is not altered. That is, if the changes are within certain limits, the sets of basic and nonbasic variables do not change. In that case, the solution of the altered problem can be obtained from the information contained in the final tableau. Otherwise, Eq. (6.23) cannot be used, and more iterations of the Simplex method are needed to obtain solution for the altered problem. Theorem 6.6 describes determination of the limits and the new right sides when the changes are within the limits.

Theorem 6.6 Limits on Changes in Resources Let Δ_k be the possible change in the right side b_k of the k th constraint. If Δ_k satisfies the following inequalities, then no more iterations of the Simplex method are required to obtain the solution for the altered problem and Eq. (6.23) can be used to determine changes to the cost function:

$$\max\{r_i < 0\} \leq \Delta_k \leq \min\{r_i > 0\}; \quad r_i = -\frac{b'_i}{a'_{ij}} \quad i = 1 \text{ to } m \quad (6.24)$$

where

b'_i = right side parameter for the i th constraint in the final tableau

a'_{ij} = parameters in the j th column of the final tableau; the j th column corresponds to x_j which is the slack variable for a “ \leq type” constraint, or the artificial variable for an equality, or “ \geq type” constraint

r_i = negative of the ratios of the right sides with the parameters in the j th column

Δ_k = possible change in the right side of the k th constraint; the slack or the artificial variable for the k th constraint determines the index j of the column whose elements are used in the Inequalities (6.24)

Furthermore, the new right side parameters b''_j due to a change of Δ_k in b_k are given as

$$b''_i = b'_i + \Delta_k a'_{ij}; \quad i = 1 \text{ to } m \quad (6.25)$$

Using Eq. (6.25) and the final tableau, new values for the basic variables in each row can be obtained. Equation (6.25) is applicable only if Δ_k is in the range determined by Inequalities (6.24). To determine the range, we first determine the column index j according to the rules given in Theorem 6.6. Then using the elements in the j th column, we determine the ratios $r_i = -b'_i/a'_{ij}$. The largest negative ratio r_i gives the lower limit on change Δ_k in b_k . If there is no $a'_{ij} > 0$ (i.e., there is no negative r_i), then the said ratio cannot be found, and so there is no lower bound on Δ_k , i.e., the lower limit is $-\infty$. The smallest positive ratio r_i gives the upper limit on change Δ_k in b_k . If there is no $a'_{ij} < 0$ (i.e., there is no positive r_i), then the said ratio cannot be found, and so there is no upper bound on Δ_k , i.e., the upper limit is ∞ . Example 6.17 demonstrates calculations of the ranges for the right side parameters and new values for the right side (i.e., the basic variables) for a problem with “ \leq type” constraints.

EXAMPLE 6.17 Ranges for Resource Limits—"≤ Type" Constraints

Find ranges for the right side parameters of constraints for the problem solved in Example 6.15.

Solution. The graphical solution for the problem is shown in Fig. 6-7. The final tableau for the problem is given in Table 6-18. For the first constraint, x_3 is the slack variable, and so $j = 3$ in Inequalities (6.24) for calculation of range for Δ_1 , the change to the constraint's right side. The ratios of the right side parameters with the elements in column 3, r_i of Eq. (6.24) are calculated as

$$r_i = -\frac{b'_i}{a'_{i3}} = \left\{ -\frac{1}{0.2}, -\frac{4}{0.4}, -\frac{13}{0.8} \right\} = \{-5.0, -10.0, -16.25\} \quad (a)$$

Since there is no positive r_i , there is no upper limit on Δ_1 . The lower limit is determined as the largest element among the negative ratios according to the Inequality (6.24) as:

$$\max\{-5.0, -10.0, -16.25\} \leq \Delta_1, \text{ or } -5 \leq \Delta_1 \quad (b)$$

Thus, limits for Δ_1 are $-5 \leq \Delta_1 \leq \infty$ and the range on b_1 is obtained by adding the current value of $b_1 = 9$ to both sides, as

$$-5 + 9 \leq \Delta_1 \leq \infty + 9, \text{ or } 4 \leq b_1 \leq \infty \quad (c)$$

For the second constraint ($k = 2$), x_4 is the slack variable. Therefore, we will use elements in column x_4 of the final tableau (a'_{i4} , $j = 4$) in the inequalities of Eq. (6.24). The ratios of the right side parameters with the elements in column 4, r_i of Eq. (6.24), are calculated as

$$r_i = -\frac{b'_i}{a'_{i4}} = \left\{ -\frac{1}{-0.4}, -\frac{4}{0.2}, -\frac{13}{1.4} \right\} = \{2.5, -20.0, -9.286\} \quad (d)$$

According to the inequalities in Eq. (6.24), lower and upper limits for Δ_2 are given as

$$\max\{-20, -9.286\} \leq \Delta_2 \leq \min\{2.5\}, \text{ or } -9.286 \leq \Delta_2 \leq 2.5 \quad (e)$$

Therefore, the allowed decrease in b_2 is 9.286 and the allowed increase is 2.5. Adding 2 to the above inequality (the current value of b_2), the range on b_2 is given as

$$-7.286 \leq b_2 \leq 4.5 \quad (f)$$

Similarly, for the third constraint, the ranges for Δ_3 and b_3 are:

$$-13 \leq \Delta_3 \leq \infty, -10 \leq b_3 \leq \infty \quad (g)$$

New values of design variables. Let us calculate new values for the design variables if the right side of the first constraint is changed from 9 to 10. Note that this change is within the limits determined in the foregoing. In Eq. (6.25), $k = 1$, so $\Delta_1 =$

$10 - 9 = 1$. Also, $j = 3$, so we use the third column from Table 6-18 in Eq. (6.25) and obtain new values of the variables as

$$x_2 = b_1'' = b_1' + \Delta_1 a'_{13} = 1 + (1)(0.2) = 1.2 \quad (\text{h})$$

$$x_1 = b_2'' = b_2' + \Delta_1 a'_{23} = 4 + (1)(0.4) = 4.4 \quad (\text{i})$$

$$x_5 = b_3'' = b_3' + \Delta_1 a'_{33} = 13 + (1)(0.8) = 13.8 \quad (\text{j})$$

The other variables remain nonbasic, so they have zero values. The new solution corresponds to point F in Fig. 6-7. Similarly, if the right side of the second constraint is changed from 2 to 3, the new values of the variables, using Eq. (6.25) and the x_4 column from Table 6-18, are calculated as $x_2 = 0.6$, $x_1 = 4.2$, and $x_5 = 14.4$. This solution corresponds to point G in Fig. 6-7.

When the right sides of two or more constraints are changed simultaneously, we can use Eq. (6.25) to determine new values of the design variables. However, we have to make sure that the new right sides do not change the basic and nonbasic sets of variables, i.e., the vertex that gives the optimum solution is not changed. Or, in other words, no new constraint becomes active. As an example, let us calculate the new values of design variables using Eq. (6.25) when the right sides of the first and the second constraints are changed to 10 and 3 from 9 and 2, respectively:

$$x_2 = b_1'' = b_1' + \Delta_1 a'_{13} + \Delta_2 a'_{14} = 1 + (1)(0.2) + (1)(-0.4) = 0.8 \quad (\text{k})$$

$$x_1 = b_2'' = b_2' + \Delta_1 a'_{23} + \Delta_2 a'_{24} = 4 + (1)(0.4) + (1)(0.2) = 4.6 \quad (\text{l})$$

$$x_5 = b_3'' = b_3' + \Delta_1 a'_{33} + \Delta_2 a'_{34} = 13 + (1)(0.8) + (1)(1.4) = 15.2 \quad (\text{m})$$

It can be verified that the new solution corresponds to point H in Fig. 6-7.

Example 6.18 demonstrates calculations of the ranges for the right side parameters and the new values for the right sides (i.e., the basic variables) for a problem with equality and “ \geq type” constraints.

EXAMPLE 6.18 Ranges for Resource Limits—Equality and “ \geq Type” Constraints

Find ranges for the right side parameters of the problem solved in Example 6.16.

Solution. The final tableau for the problem is given in Table 6-19. The graphical solution for the problem is given in Fig. 6-8. In the tableau, x_3 is a slack variable for the first constraint, x_4 is a surplus variable for the third constraint, x_5 is an artificial variable for the second constraint, and x_6 is an artificial variable for the third constraint. For the first constraint, x_3 is the slack variable, and therefore, index j for use in Inequalities (6.24) is determined as 3. Using the same procedure as for Example 6.17, the ranges for Δ_1 and b_1 are calculated as $-2 \leq \Delta_1 \leq \infty$ and $3 \leq b_1 \leq \infty$.

Since the second constraint is an equality, the index j for use in Eq. (6.24) is determined by the artificial variable x_5 for the constraint, i.e., $j = 5$. Accordingly, ratios r_i in Eq. (6.24) and the range for Δ_2 are calculated as

$$r_i = -\frac{b'_i}{a'_{i5}} = \left\{ -\frac{2}{-1}, -\frac{2/3}{1/3}, -\frac{5/3}{1/3} \right\} = \{2.0, -2.0, -5.0\} \quad (a)$$

$$\max\{-2.0, -5.0\} \leq \Delta_2 \leq \min\{2.0\}, \text{ or, } -2 \leq \Delta_2 \leq 2 \quad (b)$$

The range for b_2 can be found by adding the current value of $b_2 = 4$ to both sides of the above inequality as $2 \leq b_2 \leq 6$.

The third constraint is a " \geq type", so index j for use in Inequalities (6.24) is determined by its artificial variable x_6 ; i.e., $j = 6$. Accordingly, ratios r_i in Eq. (6.24) and the range for Δ_3 are calculated as

$$r_i = -\frac{b'_i}{a'_{i6}} = \left\{ -\frac{2}{1}, -\frac{2/3}{-2/3}, -\frac{5/3}{1/3} \right\} = \{-2.0, 1.0, -5.0\} \quad (c)$$

$$\max\{-2.0, -5.0\} \leq \Delta_3 \leq \min\{1.0\}, \text{ or } -2 \leq \Delta_3 \leq 1 \quad (d)$$

The limits on changes in b_3 are (add current value of $b_3 = 1$ to both sides of the above inequality) $-1 \leq b_3 \leq 2$.

New values of design variables. We can use Eq. (6.25) to calculate the new values of the design variables for the right side changes that remain within the previously determined ranges. It can be seen that since the first constraint is not active, it does not affect the optimum solution as long as its right side remains within the range $3 \leq b_1 \leq \infty$ determined previously.

Let us determine the new solution when the right side of the second constraint is changed to 5 from 4 (the change is within the range determined previously). The second constraint has x_5 as an artificial variable, so we use column 5 ($j = 5$) from Table 6-19 in Eq. (6.25) and obtain the new values of the variables as follows:

$$x_3 = b''_1 = b'_1 + \Delta_2 a'_{15} = 2 + (1)(-1) = 1 \quad (e)$$

$$x_2 = b''_2 = b'_2 + \Delta_2 a'_{25} = \frac{2}{3} + (1)\left(\frac{1}{3}\right) = 1 \quad (f)$$

$$x_1 = b''_3 = b'_3 + \Delta_2 a'_{35} = \frac{5}{3} + (1)\left(\frac{1}{3}\right) = 2 \quad (g)$$

To determine the new values of design variables when the right side of the third constraint is changed from 1 to 2, we use the x_6 column ($j = 6$) from Table 6-19 in Eq. (6.25) and obtain the new solution as

$$x_3 = b''_1 = b'_1 + \Delta_3 a'_{16} = 2 + (1)(1) = 3 \quad (h)$$

$$x_2 = b''_2 = b'_2 + \Delta_3 a'_{26} = \frac{2}{3} + (1)\left(-\frac{2}{3}\right) = 0 \quad (i)$$

$$x_1 = b''_3 = b'_3 + \Delta_3 a'_{36} = \frac{5}{3} + (1)\left(\frac{1}{3}\right) = 2 \quad (j)$$

It can easily be seen from Fig. 6-8 that the new solution corresponds to point C.

6.5.3 Ranging Cost Coefficients

If a cost coefficient c_k is changed to $c_k + \Delta c_k$, we like to find an admissible range on Δc_k such that the optimum design variables are not changed. *Note that when the cost coefficients are changed, the feasible region for the problem does not change.* However, orientation of the cost function hyperplane and value of the cost function change. Limits on the change Δc_k for the coefficient c_k depend on whether x_k is a basic variable at the optimum. Thus, we must consider the two cases separately. Theorems 6.7 and 6.8 give ranges for the cost coefficients for the two cases.

Theorem 6.7 Range for Cost Coefficient of Nonbasic Variables Let c_k be such that x_k^* is not a basic variable. If this c_k is replaced by any $c_k + \Delta c_k$, where $-\infty < \Delta c_k < \infty$, then the optimum solution (design variables and the cost function) does not change. Here, c'_k is the reduced cost coefficient corresponding to x_k^* in the final tableau.

Theorem 6.8 Range for Cost Coefficient of Basic Variables Let c_k be such that x_k^* is a basic variable, and let $x_k^* = b'_r$ (a superscript * is used to indicate optimum value). Then, the range for the change Δc_k in c_k for which the optimum design variables do not change is given as

$$\max\{d_j < 0\} \leq \Delta c_k \leq \min\{d_j > 0\}, \quad d_j = \frac{c'_j}{a'_{rj}} \quad (6.26)$$

where

a'_{rj} = element in the r th row and the j th column of the final tableau. The index r is determined by the row that determines x_k^* . Index j corresponds to each of the nonbasic columns excluding artificial columns. (*Note:* if no $a'_{rj} > 0$, then there is no upper limit; if no $a'_{rj} < 0$, then there is no lower limit.)

c'_j = reduced cost coefficient in the j th nonbasic column excluding artificial variable columns

d_j = ratios of the reduced cost coefficients with the elements in the r th row corresponding to nonbasic columns excluding artificial columns

When Δc_k satisfies Inequality (6.26), the optimum value of the cost function is $f^* + \Delta c_k x_k^*$.

To determine possible changes in the cost coefficient of a basic variable, the first step is to determine the row index r for use in Inequalities (6.26). This represents the row determining the basic variable x_k^* . After r has been determined, we determine ratios of the reduced cost coefficients and elements in the r th row according to the rules given in Theorem 6.8. The lower limit on Δc_k is determined by the maximum of the negative ratios. The upper limit is determined by the minimum of the positive ratios. Example 6.19 demonstrates the procedure for the “ \leq type” constraints and Example 6.20 demonstrates it for the equality and “ \geq type” constraints.

EXAMPLE 6.19 Ranges for Cost Coefficients—“ \leq Type” Constraints

Determine ranges for the cost coefficients of the problem solved in Example 6.15.

Solution. The final tableau for the problem is given in Table 6-18. The problem is solved as a minimization of the cost function $f = -5x_1 + 2x_2$. Therefore, we will find

ranges for the cost coefficients $c_1 = -5$ and $c_2 = 2$. Note that since both x_1 and x_2 are basic variables, Theorem 6.8 will be used.

Since the second row determines the basic variable x_1 , $r = 2$ (the row number) for use in Inequalities (6.26). Columns 3 and 4 are nonbasic; therefore $j = 3, 4$ are the column indices for use in Eq. (6.26). After calculating the ratios d_j , the range for Δc_1 is calculated as

$$d_j = \frac{c'_j}{a'_{2j}} = \left\{ \frac{1.6}{0.4}, \frac{1.8}{0.2} \right\} = \{4, 9\}; \quad -\infty \leq \Delta c_1 \leq \min\{4, 9\}; \quad \text{or} \quad -\infty \leq \Delta c_1 \leq 4 \quad (\text{a})$$

The range for c_1 is obtained by adding the current value of $c_1 = -5$ to both sides of the above inequality,

$$-\infty \leq \Delta c_1 \leq -1 \quad (\text{b})$$

Thus, if c_1 changes from -5 to -4 , the new cost function value is given as

$$f_{\text{new}}^* = f^* + \Delta c_1 x_1^* = -18 + (-4 - (-5))(4) = -14 \quad (\text{c})$$

That is, the cost function will increase by 4.

For the second cost coefficient, $r = 1$ (the row number) because the first row determines x_2 as a basic variable. After calculating the ratios d_j , the range for Δc_2 is calculated as

$$d_j = \frac{c'_j}{a'_{1j}} = \left\{ \frac{1.6}{0.2}, \frac{1.8}{-0.4} \right\} = \{8, -4.5\}; \quad \max\{-4.5\} \leq \Delta c_2 \leq \min\{8\};$$

$$\text{or} \quad -4.5 \leq \Delta c_2 \leq 8 \quad (\text{d})$$

The range for c_2 is obtained by adding the current value of $c_2 = 2$ to both sides of the above inequality,

$$-2.5 \leq c_2 \leq 10 \quad (\text{e})$$

Thus, if c_2 is changed from 2 to 3, the new cost function value is given as

$$f_{\text{new}}^* = f^* + \Delta c_2 x_2^* = -18 + (3 - 2)(1) = -17 \quad (\text{f})$$

Note that the range for the coefficients of the maximization function ($z = 5x_1 - 2x_2$) can be obtained from Eqs. (b) and (e). To determine these ranges, we multiply Eqs. (b) and (e) by -1 . Therefore, the range for $c_1 = 5$ is given as $1 \leq c_1 \leq \infty$, and that for $c_2 = -2$ is $-10 \leq c_2 \leq 2.5$.

EXAMPLE 6.20 Ranges for Cost Coefficients—Equality and “ \geq Type” Constraints

Find ranges for the cost coefficients of the problem solved in Example 6.16.

Solution. The final tableau for the problem is given in Table 6-19. In the tableau, x_3 is a slack variable for the first constraint, x_4 is a surplus variable for the third constraint, and x_5 and x_6 are artificial variables for the second and third constraints,

respectively. Since both x_1 and x_2 are basic variables, we will use Theorem 6.8 to find ranges for the cost coefficients $c_1 = -1$ and $c_2 = -4$. Note that the problem is solved as minimization of the cost function $f = -x_1 - 4x_2$. Columns 4, 5, and 6 are nonbasic. However, since artificial columns 5 and 6 must be excluded, only column 4 can be used in Eq. (6.26).

To find the range for Δc_1 , $r = 3$ is used because the third row determines x_1 as a basic variable. Using Inequalities (6.26) with $r = 3$ and $j = 4$, we have

$$\max\left\{\frac{7}{3}/\left(-\frac{1}{3}\right)\right\} \leq \Delta c_1 \leq \infty; \text{ or } -7 \leq \Delta c_1 \leq \infty \quad (\text{a})$$

The range for c_1 is obtained by adding the current value of $c_1 = -1$ to both sides of the inequality,

$$-8 \leq c_1 \leq \infty \quad (\text{b})$$

Thus, if c_1 changes from -1 to -2 , the new cost function value is given as

$$f_{\text{new}}^* = f^* + \Delta c_1 x_1^* = -\frac{13}{3} + (-2 - (-1))\left(\frac{5}{3}\right) = -6 \quad (\text{c})$$

For the second cost coefficient, $r = 2$ because the second row determines x_2 as a basic variable. Using Eq. (6.26) with $r = 2$ and $j = 4$, the range for Δc_2 is obtained as $-\infty \leq \Delta c_2 \leq 3.5$. Thus the range for c_2 with current value $c_2 = -4$ is given as $-\infty \leq c_2 \leq -0.5$. If c_2 changes from -4 to -3 , the new value of the cost function is given as

$$f_{\text{new}}^* = f^* + \Delta c_2 x_2^* = -\frac{13}{3} + (-3 - (-4))\left(\frac{2}{3}\right) = -\frac{11}{3} \quad (\text{d})$$

The ranges for coefficients of the *maximization function* ($z = x_1 + 4x_2$) are obtained by multiplying the above ranges by -1 , as

$$-\infty \leq c_1 \leq 8 \quad (-\infty \leq \Delta c_1 \leq 7) \text{ and } 0.5 \leq c_2 \leq \infty \quad (-3.5 \leq \Delta c_2 \leq \infty) \quad (\text{e})$$

*6.5.4 Changes in the Coefficient Matrix

Any change in the coefficient matrix \mathbf{A} in Eq. (6.10) changes the feasible region for the problem. This may change the optimum solution for the problem depending on whether the change is associated with a basic variable. Let a_{ij} be replaced by $a_{ij} + \Delta a_{ij}$. We shall determine limits for Δa_{ij} so that with minor computations the optimum solution for the changed problem can be obtained. We must consider the two cases: (i) when the change is associated with a nonbasic variable and (ii) when the change is associated with a basic variable. Results for these two cases are summarized in Theorems 6.9 and 6.10, respectively.

Theorem 6.9 Change Associated with a Nonbasic Variable Let j in a_{ij} be such that x_j is not a basic variable and k be the column index for the slack or artificial variable associated with the constraint of the i th row. Define a vector

$$\mathbf{c}_B = [c_{B1} \ c_{B2} \ \dots \ c_{Bm}]^T \quad (6.27)$$

where $c_{Bi} = c_j$ if $x_j^* = b_i^*$, $i = 1$ to m (i.e., the index i corresponds to the row that determines the optimum value of variable x_j). Recall that m is the number of constraints. Also define a scalar

$$R = \sum_{r=1}^m c_{Br} a'_{rk} \quad (6.28)$$

With this notation, if Δa_{ij} satisfies one of the following sets of inequalities, then the optimum solution (design variables and cost function) does not change when a_{ij} is replaced by any $a_{ij} + \Delta a_{ij}$:

$$\Delta a_{ij} \geq c'_j / R \text{ when } R > 0, \text{ and } \Delta a_{ij} \leq \infty \text{ when } R = 0 \quad (6.29)$$

or,

$$\Delta a_{ij} \leq c'_j / R \text{ when } R < 0, \text{ and } \Delta a_{ij} \geq -\infty \text{ when } R = 0 \quad (6.30)$$

Also, if $R = 0$, the solution does not change for any value of Δa_{ij} .

To use the theorem, a first step is to determine indices j and k . Then we determine the vector \mathbf{c}_B of Eq. (6.27), and the scalar R of Eq. (6.28). Conditions of Inequalities (6.29) and (6.30) then determine whether the given Δa_{ij} will change the optimum solution. If the inequalities are not satisfied, then we have to re-solve the problem to obtain the new solution.

Theorem 6.10 Change Associated with a Basic Variable Let j in a_{ij} be such that x_j is a basic variable and let $x_t^* = b'_t$ (i.e., t is the row index that determines optimum value of x_j). Let the index k and the scalar R be defined as in Theorem 6.9. Let Δa_{ij} satisfy the following inequalities:

$$\max_{r \neq t} \{b'_r / A_r, A_r < 0\} \leq \Delta a_{ij} \leq \min_{r \neq t} \{b'_r / A_r, A_r > 0\} \quad (6.31)$$

$$A_r = b'_r a'_{rk} - b'_t a'_{tk}, \quad r = 1 \text{ to } m; \quad r \neq t \quad (6.32)$$

and

$$\max_q \{-c'_q / B_q, B_q > 0\} \leq \Delta a_{ij} \leq \min_q \{-c'_q / B_q, B_q < 0\} \quad (6.33)$$

$$B_q = (c'_q a'_{ik} + a'_{iq} R) \text{ for all } q \text{ not in the basis} \quad (6.34)$$

and

$$1 + a'_{tk} \Delta a_{ij} > 0 \quad (6.35)$$

Note that the upper and lower limits on Δa_{ij} do not exist if the corresponding denominators in Eqs. (6.31) and (6.33) do not exist. If Δa_{ij} satisfies the above inequalities, then the optimum solution of the changed problem can be obtained without any further iterations of the Simplex method. If b'_r for $r = 1$ to m is replaced by

$$\begin{aligned} b''_r &= b'_r - \Delta a_{ij} a'_{rk} / (1 + \Delta a_{ij} a'_{tk}), \quad r = 1 \text{ to } m; \quad r \neq t \\ b''_t &= b'_t / (1 + \Delta a_{ij} a'_{tk}) \end{aligned} \quad (6.36)$$

in the final tableau, then the new optimum values for the basic variables can be obtained when a_{ij} is replaced by $a_{ij} + \Delta a_{ij}$. In other words, if $x_j^* = b'_r$, then $x'_j = b''_r$, where x'_j refers to the optimum solution for the changed problem.

To use the theorem, we need to determine indices j , t , and k . Then we determine the constants A_r and B_q from Eqs. (6.32) and (6.34). With these, ranges on Δa_{ij} can be determined from Inequalities (6.31) and (6.33). If Δa_{ij} satisfy these inequalities, Eq. (6.36) determines the new solution. If the inequalities are not satisfied, the problem must be re-solved for the new solution.

6.6 Solution of LP Problems Using Excel Solver

Excel Solver can be used to solve linear programming problems. The procedure for solving the problem is basically the same as explained for the solution of nonlinear equations in Chapter 4. An Excel worksheet needs to be prepared to enter all the data and equations for the problem. Then, the Solver dialog box is activated under the Tools menu. There, the objective function, design variables, and the constraints are defined, and the problem is solved. We shall demonstrate this process by solving the problem given in Example 6.16.

The Excel worksheet for the problem can be organized in many different ways. Figure 6-9 shows one possible format for setting up the problem. The original problem is entered at the top of the sheet just as a reminder. Other cells containing data about the problem are explained as follows:

- A10 to A15: row designations
- C11, D11: starting values for the variables x_1 and x_2 , respectively; currently set to 0
- C12, D12: objective function coefficients for x_1 and x_2 .
- C13 to D15: constraint coefficient matrix
- E12: formula to calculate the objective function value using the design variable values in the cells C11 and D11
- E13: formula to calculate the left side of constraint 1
- E14: formula to calculate the left side of constraint 2

	A	B	C	D	E	F
1	Example 6.16 LP Problem					
2						
3	Problem is to maximize:	x_1+4x_2				
4	subject to	$x_1+2x_2 \leq 5$				
5		$2x_1+x_2=4$				
6		$x_1-x_2 \geq 1$				
7		$x_1, x_2 \geq 0$				
8						
9	Problem set up for Solver					
10	Variables		x_1	x_2	Sum of LHS	RHS Limit
11	Variable value		0	0		
12	Objective function: max		1	4	$=C12*CC$11+D12*DD11	
13	Constraint 1		1	2	$=C13*CC$11+D13*DD11	5
14	Constraint 2		2	1	$=C14*CC$11+D14*DD11	4
15	Constraint 3		1	-1	$=C15*CC$11+D15*DD11	1

FIGURE 6-9 Excel worksheet for the problem of Example 6.16.

E15: formula to calculate the left side of constraint 3
 F13 to F15: right side limits for the constraints

The “Formula Auditing” command under the Tools menu is used to display the formulas in cells E12 to E15; without that command the cells display the current evaluation of the formulas. The formula in cell E12 is entered and the rest of the formulas are generated using the “copy cell” facility. It is important to note the “\$” signs used in referencing some of the cell in the formulas entered in cell E12, as “=C12*\$C\$11+D12*\$D\$11.” The cells that are required to remain fixed in the formula while copying, need to have a “\$” sign with them. For example, cells C11 and D11 have design variable values that are needed with each formula; therefore these cells are entered as \$C\$11 and \$D\$11. These cell references do not change in the formulas in cells E13 to E14. Alternatively, equations can be entered manually in each cell.

The next step is to identify the objective function, variables, and constraints for Solver. This is done by invoking Solver under the Tools menu. This is shown in Fig. 6-10 where cell E12 is identified as the objective function in the Target Cell. The “Max” button is selected to indicate that the objective is to be maximized. Next, the design variables are entered as cells C11 and D11 in the “By Changing Cells” text box. Excel will change the values in these cells as it determines the optimal solution. The constraints are entered by clicking the “Add” button; a dialog box appears in which the cells for the left and right sides of a constraint are entered. The final set-up for the present problem in the Solver Parameters box is shown in Fig. 6-10. Now we click the Options button and identify the problem as a “Linear Model” and click the Solve button to obtain Solver results.

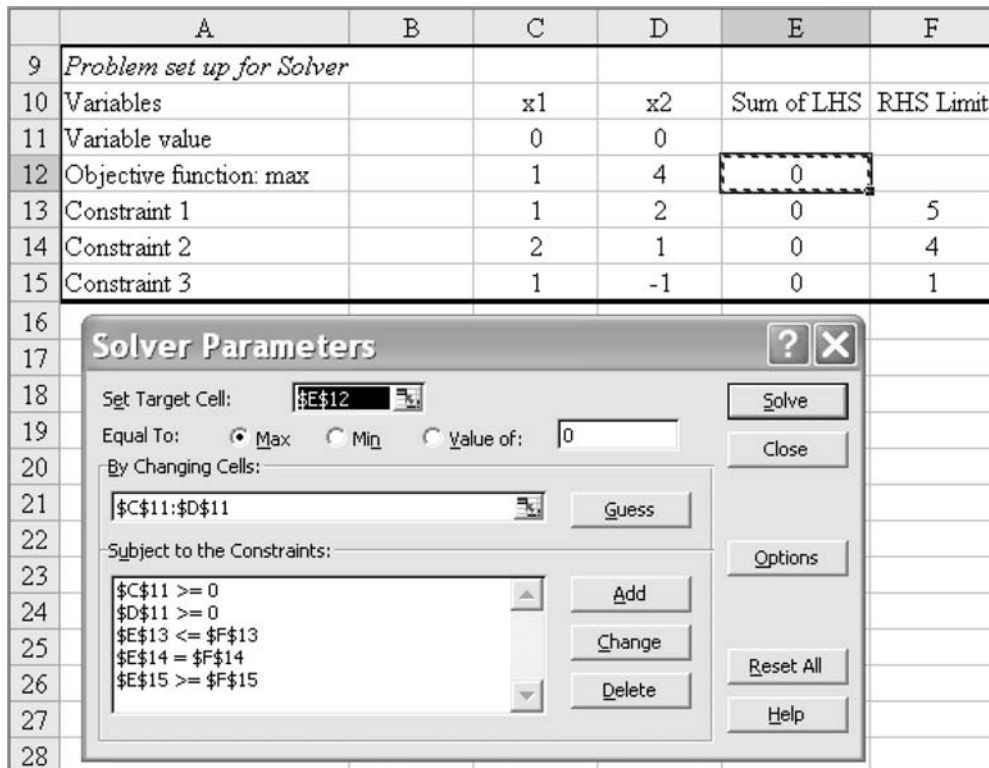


FIGURE 6-10 Solver Parameters dialog box for the problem of Example 6.16.

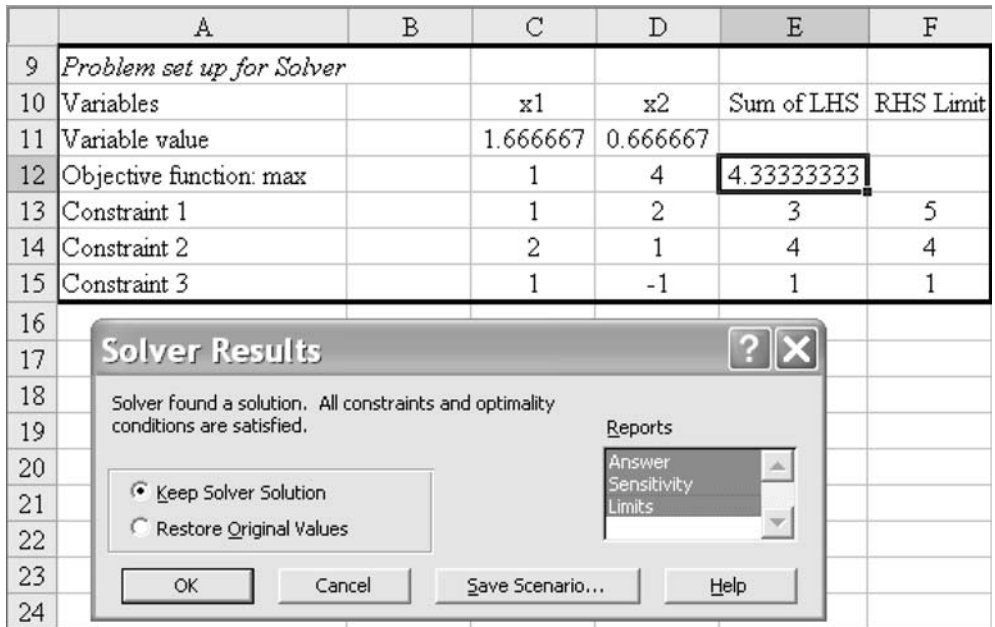


FIGURE 6-11 Solver Results dialog box for the problem of Example 6.16.

Microsoft Excel 10.0 Answer Report

Worksheet: [Example 6.16.xls]Example 6.16

Target Cell (Max)

Cell	Name	Original Value	Final Value
\$E\$12	Objective function: max Sum	0	4.33333333

Adjustable Cells

Cell	Name	Original Value	Final Value
\$C\$11	Variable value x1	0	1.66666667
\$D\$11	Variable value x2	0	0.66666667

Constraints

Cell	Name	Cell Value	Formula	Status	Slack
\$E\$13	Constraint 1 Sum of LHS	3	\$E\$13<=\$F\$13	Not Binding	2
\$E\$15	Constraint 3 Sum of LHS	1	\$E\$15>=\$F\$15	Binding	0
\$E\$14	Constraint 2 Sum of LHS	4	\$E\$14=\$F\$14	Not Binding	0
\$C\$11	Variable value x1	1.66666667	\$C\$11>=0	Not Binding	1.66666667
\$D\$11	Variable value x2	0.66666667	\$D\$11>=0	Not Binding	0.66666667

FIGURE 6-12 Answer Report from Solver for Example 6.16.

Figure 6-11 shows the Solver Results dialog box and the updated worksheet. Since the “Keep Solver Solution” option is chosen, the Solver updates the values of the cells C11, D11, and E12 to E15. Three reports are produced in separate worksheets, “Answers, Sensitivity, and Limits.” Any of these can be highlighted before clicking “OK.” Figure 6-12 shows the

Microsoft Excel 10.0 Sensitivity Report
Worksheet: [Example 6.16.xls]Example 6.16

Adjustable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
\$C\$11	Variable value x1	1.666667	0	1	7	1E+30
\$D\$11	Variable value x2	0.666667	0	4	1E+30	3.5

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
\$E\$13	Constraint 1 Sum of LHS	3	0	5	1E+30	2
\$E\$15	Constraint 3 Sum of LHS	1	-2.333333	1	1	2
\$E\$14	Constraint 2 Sum of LHS	4	1.666667	4	2	2

FIGURE 6-13 Sensitivity Report from Solver for Example 6.16.

Answer Report; it is seen that the solution obtained is the same as that given in Table 6-19. Figure 6-13 shows the Sensitivity Report for the problem. It gives ranges for the right side parameters and the objective function coefficients. It is seen that these ranges match with the values obtained for Examples 6.18 and 6.20. The Limits Report (not shown) gives the lower and upper limits for each variable and the corresponding value of the objective function. Solver determines these limits by rerunning the optimizer with all variables fixed to their optimal values except one which is optimized.

Exercises for Chapter 6

Section 6.1 Definition of Standard Linear Programming Problem

6.1 *Answer True or False.*

1. A linear programming problem having maximization of a function cannot be transcribed into the standard LP form.
2. A surplus variable must be added to a “ \leq type” constraint in the standard LP formulation.
3. A slack variable for an LP constraint can have a negative value.
4. A surplus variable for an LP constraint must be nonnegative.
5. If a “ \leq type” constraint is active, its slack variable must be positive.
6. If a “ \geq type” constraint is active, its surplus variable must be zero.
7. In the standard LP formulation, the resource limits are free in sign.
8. Only “ \leq type” constraints can be transcribed into the standard LP form.
9. Variables that are free in sign can be treated in any LP problem.
10. In the standard LP form, all the cost coefficients must be positive.
11. All variables must be nonnegative in the standard LP definition.

Convert the following problems to the standard LP form.

- 6.2 Minimize $f = 5x_1 + 4x_2 - x_3$
subject to $x_1 + 2x_2 - x_3 \geq 1$
 $2x_1 + x_2 + x_3 \geq 4$
 $x_1, x_2 \geq 0$; x_3 is unrestricted in sign
- 6.3 Maximize $z = x_1 + 2x_2$
subject to $-x_1 + 3x_2 \leq 10$
 $x_1 + x_2 \leq 6$
 $x_1 - x_2 \leq 2$
 $x_1 + 3x_2 \geq 6$
 $x_1, x_2 \geq 0$
- 6.4 Minimize $f = 2x_1 - 3x_2$
subject to $x_1 + x_2 \leq 1$
 $-2x_1 + x_2 \geq 2$
 $x_1, x_2 \geq 0$
- 6.5 Maximize $z = 4x_1 + 2x_2$
subject to $-2x_1 + x_2 \leq 4$
 $x_1 + 2x_2 \geq 2$
 $x_1, x_2 \geq 0$
- 6.6 Maximize $z = x_1 + 4x_2$
subject to $x_1 + 2x_2 \leq 5$
 $x_1 + x_2 = 4$
 $x_1 - x_2 \geq 3$
 $x_1, x_2 \geq 0$
- 6.7 Maximize $z = x_1 + 4x_2$
subject to $x_1 + 2x_2 \leq 5$
 $2x_1 + x_2 = 4$
 $x_1 - x_2 \geq 1$
 $x_1, x_2 \geq 0$
- 6.8 Minimize $f = 9x_1 + 2x_2 + 3x_3$
subject to $-2x_1 - x_2 + 3x_3 \leq -5$
 $x_1 - 2x_2 + 2x_3 \geq -2$
 $x_1, x_2, x_3 \geq 0$
- 6.9 Minimize $f = 5x_1 + 4x_2 - x_3$
subject to $x_1 + 2x_2 - x_3 \geq 1$
 $2x_1 + x_2 + x_3 \geq 4$
 $x_1, x_2 \geq 0$; x_3 is unrestricted in sign
- 6.10 Maximize $z = -10x_1 - 18x_2$
subject to $x_1 - 3x_2 \leq -3$
 $2x_1 + 2x_2 \geq 5$
 $x_1, x_2 \geq 0$
- 6.11 Minimize $f = 20x_1 - 6x_2$
subject to $3x_1 - x_2 \geq 3$

$$-4x_1 + 3x_2 = -8$$

$$x_1, x_2 \geq 0$$

6.12 Maximize $z = 2x_1 + 5x_2 - 4.5x_3 + 1.5x_4$
 subject to $5x_1 + 3x_2 + 1.5x_3 \leq 8$
 $1.8x_1 - 6x_2 + 4x_3 + x_4 \geq 3$
 $-3.6x_1 + 8.2x_2 + 7.5x_3 + 5x_4 = 15$
 $x_i \geq 0; i = 1 \text{ to } 4$

6.13 Minimize $f = 8x_1 - 3x_2 + 15x_3$
 subject to $5x_1 - 1.8x_2 - 3.6x_3 \geq 2$
 $3x_1 + 6x_2 + 8.2x_3 \geq 5$
 $1.5x_1 - 4x_2 + 7.5x_3 \geq -4.5$
 $-x_2 + 5x_3 \geq 1.5$
 $x_1, x_2 \geq 0; x_3 \text{ is unrestricted in sign}$

6.14 Maximize $z = 10x_1 + 6x_2$
 subject to $2x_1 + 3x_2 \leq 90$
 $4x_1 + 2x_2 \leq 80$
 $x_2 \geq 15$
 $5x_1 + x_2 = 25$
 $x_1, x_2 \geq 0$

6.15 Maximize $z = -2x_1 + 4x_2$
 subject to $2x_1 + x_2 \geq 3$
 $2x_1 + 10x_2 \leq 18$
 $x_1, x_2 \geq 0$

6.16 Maximize $z = x_1 + 4x_2$
 subject to $x_1 + 2x_2 \leq 5$
 $2x_1 + x_2 = 4$
 $x_1 - x_2 \geq 3$
 $x_1 \geq 0, x_2 \text{ is unrestricted in sign}$

6.17 Minimize $f = 3x_1 + 2x_2$
 subject to $x_1 - x_2 \geq 0$
 $x_1 + x_2 \geq 2$
 $x_1, x_2 \geq 0$

6.18 Maximize $z = 3x_1 + 2x_2$
 subject to $x_1 - x_2 \geq 0$
 $x_1 + x_2 \geq 2$
 $2x_1 + x_2 \leq 6$
 $x_1, x_2 \geq 0$

6.19 Maximize $z = x_1 + 2x_2$
 subject to $3x_1 + 4x_2 \leq 12$
 $x_1 + 3x_2 \geq 3$
 $x_1 \geq 0; x_2 \text{ is unrestricted in sign}$

Section 6.2 Basic Concepts Related to Linear Programming Problems

6.20 Answer True or False.

1. In the standard LP definition, the number of constraint equations (i.e., rows in the matrix \mathbf{A}) must be less than the number of variables.
2. In an LP problem, the number of “ \leq type” constraints cannot be more than the number of design variables.
3. In an LP problem, the number of “ \geq type” constraints cannot be more than the number of design variables.
4. An LP problem has an infinite number of basic solutions.
5. A basic solution must have zero value for some of the variables.
6. A basic solution can have negative values for some of the variables.
7. A degenerate basic solution has exactly m variables with nonzero values, where m is the number of equations.
8. A basic feasible solution has all variables with nonnegative values.
9. A basic feasible solution must have m variables with positive values, where m is the number of equations.
10. The optimum point for an LP problem can be inside the feasible region.
11. The optimum point for an LP problem lies at a vertex of the feasible region.
12. The solution to any LP problem is only a local optimum.
13. The solution to any LP problem is a unique global optimum.

Find all the basic solutions for the following LP problems using the Gauss-Jordan elimination method. Identify basic feasible solutions and show them on graph paper.

6.21 Maximize $z = x_1 + 4x_2$
subject to $x_1 + 2x_2 \leq 5$
 $2x_1 + x_2 = 4$
 $x_1 - x_2 \geq 1$
 $x_1, x_2 \geq 0$

6.22 Maximize $z = -10x_1 - 18x_2$
subject to $x_1 - 3x_2 \leq -3$
 $2x_1 + 2x_2 \geq 5$
 $x_1, x_2 \geq 0$

6.23 Maximize $z = x_1 + 2x_2$
subject to $3x_1 + 4x_2 \leq 12$
 $x_1 + 3x_2 \geq 3$
 $x_1 \geq 0, x_2$ is unrestricted in sign

6.24 Minimize $f = 20x_1 - 6x_2$
subject to $3x_1 - x_2 \geq 3$
 $-4x_1 + 3x_2 = -8$
 $x_1, x_2 \geq 0$

6.25 Maximize $z = 5x_1 - 2x_2$
subject to $2x_1 + x_2 \leq 9$
 $x_1 - 2x_2 \leq 2$
 $-3x_1 + 2x_2 \leq 3$
 $x_1, x_2 \geq 0$

- 6.26 Maximize $z = x_1 + 4x_2$
 subject to $x_1 + 2x_2 \leq 5$
 $x_1 + x_2 = 4$
 $x_1 - x_2 \geq 3$
 $x_1, x_2 \geq 0$
- 6.27 Minimize $f = 5x_1 + 4x_2 - x_3$
 subject to $x_1 + 2x_2 - x_3 \geq 1$
 $2x_1 + x_2 + x_3 \geq 4$
 $x_1, x_3 \geq 0$; x_2 is unrestricted in sign
- 6.28 Minimize $f = 9x_1 + 2x_2 + 3x_3$
 subject to $-2x_1 - x_2 + 3x_3 \leq -5$
 $x_1 - 2x_2 + 2x_3 \geq -2$
 $x_1, x_2, x_3 \geq 0$
- 6.29 Maximize $z = 4x_1 + 2x_2$
 subject to $-2x_1 + x_2 \leq 4$
 $x_1 + 2x_2 \geq 2$
 $x_1, x_2 \geq 0$
- 6.30 Maximize $z = 3x_1 + 2x_2$
 subject to $x_1 - x_2 \geq 0$
 $x_1 + x_2 \geq 2$
 $x_1, x_2 \geq 0$
- 6.31 Maximize $z = 4x_1 + 5x_2$
 subject to $-x_1 + 2x_2 \leq 10$
 $3x_1 + 2x_2 \leq 18$
 $x_1, x_2 \geq 0$

Section 6.3 Basic Ideas and Concepts of the Simplex Method

Solve the following problems by the Simplex method and verify the solution graphically whenever possible.

- 6.32 Maximize $z = x_1 + 0.5x_2$
 subject to $6x_1 + 5x_2 \leq 30$
 $3x_1 + x_2 \leq 12$
 $x_1 + 3x_2 \leq 12$
 $x_1, x_2 \geq 0$
- 6.33 Maximize $z = 3x_1 + 2x_2$
 subject to $3x_1 + 2x_2 \leq 6$
 $-4x_1 + 9x_2 \leq 36$
 $x_1, x_2 \geq 0$
- 6.34 Maximize $z = x_1 + 2x_2$
 subject to $-x_1 + 3x_2 \leq 10$
 $x_1 + x_2 \leq 6$
 $x_1 - x_2 \leq 2$
 $x_1, x_2 \geq 0$

- 6.35 Maximize $z = 2x_1 + x_2$
subject to $-x_1 + 2x_2 \leq 10$
 $3x_1 + 2x_2 \leq 18$
 $x_1, x_2 \geq 0$
- 6.36 Maximize $z = 5x_1 - 2x_2$
subject to $2x_1 + x_2 \leq 9$
 $x_1 - x_2 \leq 2$
 $-3x_1 + 2x_2 \leq 3$
 $x_1, x_2 \geq 0$
- 6.37 Minimize $f = 2x_1 - x_2$
subject to $-x_1 + 2x_2 \leq 10$
 $3x_1 + 2x_2 \leq 18$
 $x_1, x_2 \geq 0$
- 6.38 Minimize $f = -x_1 + x_2$
subject to $2x_1 + x_2 \leq 4$
 $-x_1 - 2x_2 \geq -4$
 $x_1, x_2 \geq 0$
- 6.39 Maximize $z = 2x_1 - x_2$
subject to $x_1 + 2x_2 \leq 6$
 $2 \geq x_1$
 $x_1, x_2 \geq 0$
- 6.40 Maximize $z = x_1 + x_2$
subject to $4x_1 + 3x_2 \leq 12$
 $x_1 + 2x_2 \leq 4$
 $x_1, x_2 \geq 0$
- 6.41 Maximize $z = -2x_1 + x_2$
subject to $x_1 \leq 2$
 $x_1 + 2x_2 \leq 6$
 $x_1, x_2 \geq 0$
- 6.42 Maximize $z = 2x_1 + x_2$
subject to $4x_1 + 3x_2 \leq 12$
 $x_1 + 2x_2 \leq 4$
 $x_1, x_2 \geq 0$
- 6.43 Minimize $f = 9x_1 + 2x_2 + 3x_3$
subject to $2x_1 + x_2 - 3x_3 \geq -5$
 $x_1 - 2x_2 + 2x_3 \geq -2$
 $x_1, x_2, x_3 \geq 0$
- 6.44 Maximize $z = x_1 + x_2$
subject to $4x_1 + 3x_2 \leq 9$
 $x_1 + 2x_2 \leq 6$
 $2x_1 + x_2 \leq 6$
 $x_1, x_2 \geq 0$

- 6.45 Minimize $f = -x_1 - 4x_2$
subject to $x_1 + x_2 \leq 16$
 $x_1 + 2x_2 \leq 28$
 $24 \geq 2x_1 + x_2$
 $x_1, x_2 \geq 0$
- 6.46 Minimize $f = x_1 - x_2$
subject to $4x_1 + 3x_2 \leq 12$
 $x_1 + 2x_2 \leq 4$
 $4 \geq 2x_1 + x_2$
 $x_1, x_2 \geq 0$
- 6.47 Maximize $z = 2x_1 + 3x_2$
subject to $x_1 + x_2 \leq 16$
 $-x_1 - 2x_2 \geq -28$
 $24 \geq 2x_1 + x_2$
 $x_1, x_2 \geq 0$
- 6.48 Maximize $z = x_1 + 2x_2$
subject to $2x_1 - x_2 \geq 0$
 $2x_1 + 3x_2 \geq -6$
 $x_1, x_2 \geq 0$
- 6.49 Maximize $z = 2x_1 + 2x_2 + x_3$
subject to $10x_1 + 9x_3 \leq 375$
 $x_1 + 3x_2 + x_3 \leq 33$
 $2 \geq x_3$
 $x_1, x_2, x_3 \geq 0$
- 6.50 Maximize $z = x_1 + 2x_2$
subject to $-2x_1 - x_2 \geq -5$
 $3x_1 + 4x_2 \leq 10$
 $x_1 \leq 2$
 $x_1, x_2 \geq 0$
- 6.51 Minimize $f = -2x_1 - x_2$
subject to $-2x_1 - x_2 \geq -5$
 $3x_1 + 4x_2 \leq 10$
 $x_1 \leq 3$
 $x_1, x_2 \geq 0$
- 6.52 Maximize $z = 12x_1 + 7x_2$
subject to $2x_1 + x_2 \leq 5$
 $3x_1 + 4x_2 \leq 10$
 $x_1 \leq 2$
 $x_2 \leq 3$
 $x_1, x_2 \geq 0$
- 6.53 Maximize $z = 10x_1 + 8x_2 + 5x_3$
subject to $10x_1 + 9x_2 \leq 375$
 $5x_1 + 15x_2 + 3x_3 \leq 35$
 $3 \geq x_3$
 $x_1, x_2, x_3 \geq 0$

Section 6.4 Two Phase Simplex Method—Artificial Variables

6.54 *Answer True or False.*

1. A pivot step of the Simplex method replaces a current basic variable with a nonbasic variable.
2. The pivot step brings the design point to the interior of the constraint set.
3. The pivot column in the Simplex method is determined by the largest reduced cost coefficient corresponding to a basic variable.
4. The pivot row in the Simplex method is determined by the largest ratio of right side parameters with the positive coefficients in the pivot column.
5. The criterion for a current basic variable to leave the basic set is to keep the new solution basic and feasible.
6. A move from one basic feasible solution to another corresponds to extreme points of the convex polyhedral set.
7. A move from one basic feasible solution to another can increase the cost function value in the Simplex method.
8. The right sides in the Simplex tableau can assume negative values.
9. The right sides in the Simplex tableau can become zero.
10. The reduced cost coefficients corresponding to the basic variables must be positive at the optimum.
11. If a reduced cost coefficient corresponding to a nonbasic variable is zero at the optimum point, there may be multiple solutions to the problem.
12. If all elements in the pivot column are negative, the problem is infeasible.
13. The artificial variables must be positive in the final solution.
14. If artificial variables are positive at the final solution, the artificial cost function is also positive.
15. If artificial cost function is positive at the optimum solution, the problem is unbounded.

Solve the following LP problems by the Simplex method and verify the solution graphically, whenever possible.

6.55 Maximize $z = x_1 + 2x_2$
 subject to $-x_1 + 3x_2 \leq 10$
 $x_1 + x_2 \leq 6$
 $x_1 - x_2 \leq 2$
 $x_1 + 3x_2 \geq 6$
 $x_1, x_2 \geq 0$

6.56 Maximize $z = 4x_1 + 2x_2$
 subject to $-2x_1 + x_2 \leq 4$
 $x_1 + 2x_2 \geq 2$
 $x_1, x_2 \geq 0$

6.57 Maximize $z = x_1 + 4x_2$
 subject to $x_1 + 2x_2 \leq 5$
 $x_1 + x_2 = 4$
 $x_1 - x_2 \geq 3$
 $x_1, x_2 \geq 0$

6.58 Maximize $z = x_1 + 4x_2$
 subject to $x_1 + 2x_2 \leq 5$

- $$2x_1 + x_2 = 4$$
- $$x_1 - x_2 \geq 1$$
- $$x_1, x_2 \geq 0$$
- 6.59 Minimize $f = 3x_1 + x_2 + x_3$
subject to $-2x_1 - x_2 + 3x_3 \leq -5$
 $x_1 - 2x_2 + 3x_3 \geq -2$
 $x_1, x_2, x_3 \geq 0$
- 6.60 Minimize $f = 5x_1 + 4x_2 - x_3$
subject to $x_1 + 2x_2 - x_3 \geq 1$
 $2x_1 + x_2 + x_3 \geq 4$
 $x_1, x_2 \geq 0$; x_3 is unrestricted in sign
- 6.61 Maximize $z = -10x_1 - 18x_2$
subject to $x_1 - 3x_2 \leq -2$
 $2x_1 + 2x_2 \geq 5$
 $x_1, x_2 \geq 0$
- 6.62 Minimize $f = 20x_1 - 6x_2$
subject to $3x_1 - x_2 \geq 3$
 $-4x_1 + 3x_2 = -8$
 $x_1, x_2 \geq 0$
- 6.63 Maximize $z = 2x_1 + 5x_2 - 4.5x_3 + 1.5x_4$
subject to $5x_1 + 3x_2 + 1.5x_3 \leq 8$
 $1.8x_1 - 6x_2 + 4x_3 + x_4 \geq 3$
 $-3.6x_1 + 8.2x_2 + 7.5x_3 + 5x_4 = 15$
 $x_i \geq 0$; $i = 1$ to 4
- 6.64 Minimize $f = 8x_1 - 3x_2 + 15x_3$
subject to $5x_1 - 1.8x_2 - 3.6x_3 \geq 2$
 $3x_1 + 6x_2 + 8.2x_3 \geq 5$
 $1.5x_1 - 4x_2 + 7.5x_3 \geq -4.5$
 $-x_2 + 5x_3 \geq 1.5$
 $x_1, x_2 \geq 0$; x_3 is unrestricted in sign
- 6.65 Maximize $z = 10x_1 + 6x_2$
subject to $2x_1 + 3x_2 \leq 90$
 $4x_1 + 2x_2 \leq 80$
 $x_2 \geq 15$
 $5x_1 + x_2 = 25$
 $x_1, x_2 \geq 0$
- 6.66 Maximize $z = -2x_1 + 4x_2$
subject to $2x_1 + x_2 \geq 3$
 $2x_1 + 10x_2 \leq 18$
 $x_1, x_2 \geq 0$
- 6.67 Maximize $z = x_1 + 4x_2$
subject to $x_1 + 2x_2 \leq 5$
 $2x_1 + x_2 = 4$

- $x_1 - x_2 \geq 3$
 $x_1 \geq 0$; x_2 is unrestricted in sign
- 6.68 Minimize $f = 3x_1 + 2x_2$
subject to $x_1 - x_2 \geq 0$
 $x_1 + x_2 \geq 2$
 $x_1, x_2 \geq 0$
- 6.69 Maximize $z = 3x_1 + 2x_2$
subject to $x_1 - x_2 \geq 0$
 $x_1 + x_2 \geq 2$
 $2x_1 + x_2 \leq 6$
 $x_1, x_2 \geq 0$
- 6.70 Maximize $z = x_1 + 2x_2$
subject to $3x_1 + 4x_2 \leq 12$
 $x_1 + 3x_2 \leq 3$
 $x_1 \geq 0$; x_2 is unrestricted in sign
- 6.71 Minimize $f = x_1 + 2x_2$
subject to $-x_1 + 3x_2 \leq 20$
 $x_1 + x_2 \leq 6$
 $x_1 - x_2 \leq 12$
 $x_1 + 3x_2 \geq 6$
 $x_1, x_2 \geq 0$
- 6.72 Maximize $z = 3x_1 + 8x_2$
subject to $3x_1 + 4x_2 \leq 20$
 $x_1 + 3x_2 \geq 6$
 $x_1 \geq 0$; x_2 is unrestricted in sign
- 6.73 Minimize $f = 2x_1 - 3x_2$
subject to $x_1 + x_2 \leq 1$
 $-2x_1 + x_2 \geq 2$
 $x_1, x_2 \geq 0$
- 6.74 Minimize $f = 3x_1 - 3x_2$
subject to $-x_1 + x_2 \leq 0$
 $x_1 + x_2 \geq 2$
 $x_1, x_2 \geq 0$
- 6.75 Minimize $f = 5x_1 + 4x_2 - x_3$
subject to $x_1 + 2x_2 - x_3 \geq 1$
 $2x_1 + x_2 + x_3 \geq 4$
 $x_1, x_2 \geq 0$; x_3 is unrestricted in sign
- 6.76 Maximize $z = 4x_1 + 5x_2$
subject to $x_1 - 2x_2 \leq -10$
 $3x_1 + 2x_2 \leq 18$
 $x_1, x_2 \geq 0$
- 6.77 Formulate and solve the optimum design problem of Exercise 2.2. Verify the solution graphically.

- 6.78 Formulate and solve the optimum design problem of Exercise 2.6. Verify the solution graphically.
- 6.79 Formulate and solve the optimum design problem of Exercise 2.7. Verify the solution graphically.
- 6.80 Formulate and solve the optimum design problem of Exercise 2.8. Verify the solution graphically.
- 6.81* Formulate and solve the optimum design problem of Exercise 2.18.
- 6.82* Formulate and solve the optimum design problem of Exercise 2.20.
- 6.83 Solve the “saw mill” problem formulated in Section 2.4.
- 6.84* Formulate and solve the optimum design problem of Exercise 2.21.
- 6.85* Obtain solutions for the three formulations of the “cabinet design” problem given in Section 2.6. Compare solutions for the three formulations.

Section 6.5 Postoptimality Analysis

- 6.86 Formulate and solve the “crude oil” problem stated in Exercise 2.2. What is the effect on the cost function if the market for lubricating oil suddenly increases to 12,000 barrels? What is the effect on the solution if the price of Crude A drops to \$24/bbl. Verify the solutions graphically.
- 6.87 Formulate and solve the problem stated in Exercise 2.6. What are the effects of the following changes? Verify your solutions graphically.
 1. The supply of material C increases to 120 kg.
 2. The supply of material D increases to 100 kg.
 3. The market for product A decreases to 60.
 4. The profit for A decreases to \$8/kg.

Solve the following problems and determine Lagrange multipliers for the constraints at the optimum point.

- | | | |
|---------------------|---------------------|---------------------|
| 6.88 Exercise 6.55 | 6.89 Exercise 6.56 | 6.90 Exercise 6.57 |
| 6.91 Exercise 6.58 | 6.92 Exercise 6.59 | 6.93 Exercise 6.60 |
| 6.94 Exercise 6.61 | 6.95 Exercise 6.62 | 6.96 Exercise 6.63 |
| 6.97 Exercise 6.64 | 6.98 Exercise 6.65 | 6.99 Exercise 6.66 |
| 6.100 Exercise 6.67 | 6.101 Exercise 6.68 | 6.102 Exercise 6.69 |
| 6.103 Exercise 6.70 | 6.104 Exercise 6.71 | 6.105 Exercise 6.72 |
| 6.106 Exercise 6.73 | 6.107 Exercise 6.74 | 6.108 Exercise 6.75 |
| 6.109 Exercise 6.76 | | |

Solve the following problems and determine ranges for the right side parameters.

- | | | |
|---------------------|---------------------|---------------------|
| 6.110 Exercise 6.55 | 6.111 Exercise 6.56 | 6.112 Exercise 6.57 |
| 6.113 Exercise 6.58 | 6.114 Exercise 6.59 | 6.115 Exercise 6.60 |
| 6.116 Exercise 6.61 | 6.117 Exercise 6.62 | 6.118 Exercise 6.63 |
| 6.119 Exercise 6.64 | 6.120 Exercise 6.65 | 6.121 Exercise 6.66 |

- | | | |
|---------------------|---------------------|---------------------|
| 6.122 Exercise 6.67 | 6.123 Exercise 6.68 | 6.124 Exercise 6.69 |
| 6.125 Exercise 6.70 | 6.126 Exercise 6.71 | 6.127 Exercise 6.72 |
| 6.128 Exercise 6.73 | 6.129 Exercise 6.74 | 6.130 Exercise 6.75 |
| 6.131 Exercise 6.76 | | |

Solve the following problems and determine ranges for the coefficients of the objective function.

- | | | |
|---------------------|---------------------|---------------------|
| 6.132 Exercise 6.55 | 6.133 Exercise 6.56 | 6.134 Exercise 6.57 |
| 6.135 Exercise 6.58 | 6.136 Exercise 6.59 | 6.137 Exercise 6.60 |
| 6.138 Exercise 6.61 | 6.139 Exercise 6.62 | 6.140 Exercise 6.63 |
| 6.141 Exercise 6.64 | 6.142 Exercise 6.65 | 6.143 Exercise 6.66 |
| 6.144 Exercise 6.67 | 6.145 Exercise 6.68 | 6.146 Exercise 6.69 |
| 6.147 Exercise 6.70 | 6.148 Exercise 6.71 | 6.149 Exercise 6.72 |
| 6.150 Exercise 6.73 | 6.151 Exercise 6.74 | 6.152 Exercise 6.75 |
| 6.153 Exercise 6.76 | | |

- 6.154* Formulate and solve the optimum design problem of Exercise 2.2. Determine Lagrange multipliers for the constraints. Calculate the ranges for the right side parameters, and the coefficients of the objective function. Verify your results graphically.
- 6.155* Formulate and solve the optimum design problem of Exercise 2.6. Determine Lagrange multipliers for the constraints. Calculate the ranges for the parameters of the right side and the coefficients of the objective function. Verify your results graphically.
- 6.156 Formulate and solve the “diet” problem stated in Exercise 2.7. Investigate the effect on the optimum solution of the following changes:
1. The cost of milk increases to \$1.20/kg.
 2. The need for vitamin A increases to 6 units.
 3. The need for vitamin B decreases to 3 units.
- Verify the solution graphically.
- 6.157 Formulate and solve the problem stated in Exercise 2.8. Investigate the effect on the optimum solution of the following changes:
1. The supply of empty bottles decreases to 750.
 2. The profit on a bottle of wine decreases to \$0.80.
 3. Only 200 bottles of alcohol can be produced.
- 6.158* Formulate and solve the problem stated in Exercise 2.18. Investigate the effect on the optimum solution of the following changes:
1. The profit on margarine increases to \$0.06/kg.
 2. The supply of milk base substances increases to 2500 kg
 3. The supply of soybeans decreases to 220,000 kg.
- 6.159 Solve the “saw mill” problem formulated in Section 2.4. Investigate the effect on the optimum solution of the following changes:
1. The transportation cost for the logs increases to \$0.16 per kilometer per log.
 2. The capacity of Mill A decreases to 200 logs/day.
 3. The capacity of Mill B decreases to 270 logs/day.

- 6.160* Formulate and solve the problem stated in Exercise 2.20. Investigate the effect on the optimum solution of the following changes:
1. Due to demand on capital, the available cash decreases to \$1.8 million.
 2. The initial investment for truck B increases to \$65,000.
 3. Maintenance capacity decreases to 28 trucks.
- 6.161* Formulate and solve the “steel mill” problem stated in Exercise 2.21. Investigate the effect on the optimum solution of the following changes:
1. The capacity of reduction plant 1 increases to 1,300,000.
 2. The capacity of reduction plant 2 decrease to 950,000.
 3. The capacity of fabricating plant 2 increases to 250,000.
 4. The demand for product 2 increases to 130,000.
 5. The demand for product 1 decreases to 280,000.
- 6.162* Obtain solutions for the three formulations of the “cabinet design” problem given in Section 2.6. Compare the three formulations. Investigate the effect on the optimum solution of the following changes:
1. Bolting capacity is decreased to 5500/day.
 2. The cost of riveting the C_1 component increases to \$0.70.
 3. The company must manufacture only 95 devices per day.
- 6.163 Given the following problem:
- $$\begin{aligned} &\text{minimize } f = 2x_1 - 4x_2 \\ &\text{subject to } g_1 = 10x_1 + 5x_2 \leq 15 \\ &\quad \quad \quad g_2 = 4x_1 + 10x_2 \leq 36 \\ &\quad \quad \quad x_1 \geq 0, x_2 \geq 0 \end{aligned}$$

Slack variables for g_1 and g_2 are x_3 and x_4 , respectively. The final tableau for the problem is given in Table E6.163. Using the given tableau:

1. Determine the optimum values of f and \mathbf{x} .
2. Determine Lagrange multipliers for g_1 and g_2 .
3. Determine the ranges for the right sides of g_1 and g_2 .
4. What is the smallest value that f can have, with the current basis, if the right side of g_1 is changed? What is the right side of g_1 for that case?

TABLE E6.163 Final Tableau for Exercise 6.163

x_1	x_2	x_3	x_4	b
2	1	$\frac{1}{5}$	0	3
-16	0	-2	1	6
10	0	$\frac{4}{5}$	0	$f + 12$

7 More on Linear Programming Methods for Optimum Design

Upon completion of this chapter, you will be able to:

- Derive the Simplex method and understand the theory behind its steps
- Use an alternate form of the two-phase Simplex method called the Big-M method
- Write a dual problem for the given LP problem
- Recover solution for the original LP problem from the solution of the dual problem

In this chapter, some additional topics related to linear programming problems are presented. These topics are usually not covered in an undergraduate course on optimum design. They may also be omitted on the first independent reading of the book.

7.1 Derivation of the Simplex Method

In the previous chapter, we presented the basic ideas and concepts of the Simplex method. The steps of the method were described and illustrated in several examples. In this section, we describe the theory that leads to the steps used in the example problems.

7.1.1 Selection of a Basic Variable That Should Become Nonbasic

Derivation of the Simplex method is based on answering the two questions posed earlier: (1) which current nonbasic variable should become basic, and (2) which current basic variable should become nonbasic. We will answer the second question in this section. Assume for the moment that x_r is a nonbasic variable tapped to become basic. This indicates that the r th nonbasic column should replace some current basic column. After this interchange, there should be all zero elements in the r th column except a positive unit element at one location.

To determine a current basic variable that should become nonbasic, we need to determine the pivot row for the elimination process. This way the current basic variable associated with that row will become nonbasic after the elimination step. To determine the pivot row, we transfer all the terms associated with the current nonbasic variable x_r (tapped to become basic) to the right side of the canonical form of Eq. (6.13). The system of equations becomes:

$$\begin{aligned}
x_1 + a_{1,m+1}x_{m+1} + \dots + a_{1,n}x_n &= b_1 - a_{1,r}x_r \\
x_2 + a_{2,m+1}x_{m+1} + \dots + a_{2,n}x_n &= b_2 - a_{2,r}x_r \\
\cdot & \\
\cdot & \\
\cdot & \\
x_m + a_{m,m+1}x_{m+1} + \dots + a_{m,n}x_n &= b_m - a_{m,r}x_r
\end{aligned} \tag{7.1}$$

Since x_r is to become a basic variable, its value should become nonnegative in the new solution. The new solution must also remain feasible. The right sides of Eq. (7.1) represent values of the basic variables for the next Simplex iteration once x_r is assigned a value greater than or equal to 0. An examination of these right sides shows that x_r cannot increase arbitrarily. The reason is that if x_r becomes arbitrarily large, then some of the new right side parameters ($b_i - a_{i,r}x_r$), $i = 1$ to m may become negative. Since right side parameters are the new values of the basic variables, the new basic solution will not be feasible. Thus for the new solution to be basic and feasible, the following constraints must be satisfied by the right sides of Eq. (7.1) in selecting a current basic variable that should become nonbasic (i.e., attain zero value):

$$b_i - a_{i,r}x_r \geq 0; \quad i = 1 \text{ to } m \tag{7.2}$$

Any $a_{i,r}$ that are nonpositive pose no limit on how much x_r can be increased since Inequality (7.2) remains satisfied; recall that $b_i \geq 0$. For a positive $a_{i,r}$, x_r can be increased from zero until one of the inequalities in Eq. (7.2) becomes active, i.e., one of the right sides of Eq. (7.1) becomes zero. A further increase would violate the nonnegativity conditions of Eq. (7.2). Thus, the maximum value that the incoming variable x_r can take is given as

$$\frac{b_s}{a_{s,r}} = \min_i \left\{ \frac{b_i}{a_{i,r}}, a_{i,r} > 0; i = 1 \text{ to } m \right\} \tag{7.3}$$

where s is the index of the smallest ratio. Equation (7.3) says that we take ratios of the right side parameters b_i with the positive elements in the r th column ($a_{i,r}$'s) and we select the row index s giving the smallest ratio. *In the case of a tie, the choice for the index s is arbitrary among the tying indices and in such a case the resulting basic feasible solution may be degenerate.* Thus, Eq. (7.3) identifies a row with the smallest ratio $b_i/a_{i,r}$. The basic variable x_s associated with this row should become nonbasic. *If all $a_{i,r}$ are nonpositive in the r th column, then x_r can be increased indefinitely.* This indicates that the LP problem is *unbounded*. Any practical problem with this situation is not properly constrained so the problem formulation should be reexamined.

7.1.2 Selection of a Nonbasic Variable That Should Become Basic

We now know how to select a basic variable that should replace a nonbasic variable. To answer the first question posed earlier, let us see how we can identify the nonbasic variable that should become basic. *The main idea of bringing a nonbasic variable into the basic set is to improve the design, i.e., to reduce the current value of the cost function.* A clue to the desired improvement is obtained if we examine the cost function expression. To do this we need to write the cost function in terms of the nonbasic variables only. We substitute for the current values of basic variables from Eq. (6.13) into the cost function to eliminate the basic variables. Current values of the basic variables are given in terms of the nonbasic variables as follows:

$$x_i = b_i - \sum_{j=m+1}^n a_{ij}x_j; \quad i = 1 \text{ to } m \quad (7.4)$$

Substituting Eq. (7.4) into the cost function expression in Eq. (6.7) and simplifying, we obtain an expression for the cost function in terms of the nonbasic variables (x_j , $j = m + 1$ to n) as

$$f = f_0 + \sum_{j=m+1}^n c'_j x_j \quad (7.5)$$

where f_0 is the current value of the cost function given as

$$f_0 = \sum_{i=1}^m b_i c_i \quad (7.6)$$

and the parameters c'_j are

$$c'_j = c_j - \sum_{i=1}^m a_{ij} c_i; \quad j = (m + 1) \text{ to } n \quad (7.7)$$

The cost coefficients c'_j of the nonbasic variables play a key role in the Simplex method and are called the *reduced or relative cost coefficients*. They are used to identify a nonbasic variable that should become basic to reduce the current value of the cost function. Expressing the cost function in terms of the current nonbasic variables is a key step in the Simplex method. We have seen that this is not difficult to accomplish because the Gaussian elimination steps can be used routinely on the cost function expression to eliminate basic variables from it. Once this has been done, the reduced cost coefficients c'_j can be readily identified.

In general the reduced cost coefficients c'_j of the nonbasic variables may be positive, negative, or zero. Let one of c'_j be negative. Then, note from Eq. (7.5) that if a positive value is assigned to the associated nonbasic variable (i.e., it is made basic), the value of f will decrease. If more than one negative c'_j is present, a widely used rule of thumb is to choose the nonbasic variable associated with the smallest c'_j (i.e., negative c'_j with the largest absolute value) to become basic. Thus, if any c'_j for $(m + 1) \leq j \leq n$ (for nonbasic variables) is negative, then it is possible to find a new basic feasible solution (if one exists) that will further reduce the cost function. If a c'_j is zero, then the associated nonbasic variable can be made basic without affecting the cost function value. If all c'_j are nonnegative, then it is not possible to reduce the cost function any further and the current basic feasible solution is optimum. These results have been summarized previously in Theorems 6.3 and 6.4.

Note that when all c'_j in the nonbasic columns are positive, the optimum solution is unique. If at least one c'_j (reduced cost coefficient associated with a nonbasic variable) is zero, then there is a possibility of alternate optima. If the nonbasic variable associated with a zero reduced cost coefficient can be made basic according to the foregoing procedure, the extreme point corresponding to alternate optima can be obtained. Since the reduced cost coefficient is zero, the optimum cost function value will not change. Any point on the line segment joining the optimum extreme points also corresponds to an optimum. Note that these optima are global as opposed to local, although there is *no distinct global optimum*. Geometrically, multiple optima for an LP problem imply that the cost function hyperplane is parallel to one of the active constraint hyperplanes.

Note that if the nonbasic variable associated with the negative reduced cost coefficient c'_j cannot be made basic (e.g., when all a_{ij} in the c'_j column are negative), then the *feasible region is unbounded*.

7.2 Alternate Simplex Method

A slightly different procedure can be used to solve linear programming problems having “ \geq type” and equality constraints. The artificial variables are introduced into the problem as before. However, the artificial cost function is not used. Instead, *the original cost function is augmented by adding to it the artificial variables multiplied by large positive constants. The additional terms act as penalties for having artificial variables in the problem.* Since artificial variables are basic, they need to be eliminated from the cost function before the Simplex method can be used to solve the preceding modified problem. This can easily be done by using the appropriate equations that contain artificial variables. Once this has been done, the regular Simplex method can be used to solve the problem. We illustrate the procedure with Example 7.1.

EXAMPLE 7.1 The Big-M Method for Equality and “ \geq Type” Constraints

Find the numerical solution for the problem given in Example 6.11 using the alternate Simplex procedure: maximize $z = y_1 + 2y_2$ subject to $3y_1 + 2y_2 \leq 12$, $2y_1 + 3y_2 \geq 6$, $y_1 \geq 0$, y_2 is unrestricted in sign.

Solution. Since y_2 is unrestricted, it has been defined as $y_2 = x_2 - x_3$. Converting the problem to the standard form, we obtain:

$$\text{minimize } f = -x_1 - 2x_2 + 2x_3 \quad (\text{a})$$

subject to

$$3x_1 + 2x_2 - 2x_3 + x_4 = 12 \quad (\text{b})$$

$$2x_1 + 3x_2 - 3x_3 - x_5 + x_6 = 6 \quad (\text{c})$$

$$x_i \geq 0; \quad i = 1 \text{ to } 6 \quad (\text{d})$$

Where x_4 is a slack variable, x_5 is a surplus variable, and x_6 is an artificial variable. Following the alternate Simplex procedure, we add Mx_6 (with, say, $M = 10$) to the cost function and obtain $f = -x_1 - 2x_2 + 2x_3 + 10x_6$. Note that if there is a feasible solution to the problem, then all artificial variables will become nonbasic, i.e., zero, and we will recover the original cost function. Also note that if there are other artificial variables in the problem, they will be multiplied by M and added to the cost function. This is sometimes called the *Big-M Method*. Now substituting for x_6 from the second constraint into the foregoing cost function, we get

$$f = -x_1 - 2x_2 + 2x_3 + 10(6 - 2x_1 - 3x_2 + 3x_3 + x_5) = 60 - 21x_1 - 32x_2 + 32x_3 + 10x_5$$

This is written as $-21x_1 - 32x_2 + 32x_3 + 10x_5 = f - 60$ in the Simplex tableau. With this cost function, iterations of the Simplex method are shown in Table 7-1. It can be seen that the final solution is the same as given in Table 6-14 and Fig. 6-4.

TABLE 7-1 Solution for Example 7.1 by Alternate Simplex Method

Initial tableau: x_6 is identified to be replaced with x_2 in the basic set.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	b	Ratio
x_4	3	2	-2	1	0	0	12	$\frac{12}{2} = 6$
x_6	2	3	-3	0	-1	1	6	$\frac{6}{3} = 2$
Cost	-21	-32	32	0	10	0	$f - 60$	

Second tableau: x_4 is identified to be replaced with x_5 in the basic set.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	b	Ratio
x_4	$\frac{5}{3}$	0	0	1	$\frac{2}{3}$	$-\frac{2}{3}$	8	$\frac{8}{2/3} = 12$
x_2	$\frac{2}{3}$	1	-1	0	$-\frac{1}{3}$	$\frac{1}{3}$	2	Negative
Cost	$\frac{1}{3}$	0	0	0	$-\frac{2}{3}$	$\frac{32}{3}$	$f + 4$	

Third tableau: Reduced cost coefficients in nonbasic columns are nonnegative; the tableau gives the optimum point.

Basic↓	x_1	x_2	x_3	x_4	x_5	x_6	b
x_4	$\frac{5}{2}$	0	0	$\frac{3}{2}$	1	-1	12
x_2	$\frac{3}{2}$	1	-1	$\frac{1}{2}$	0	0	6
Cost	2	0	0	1	0	10	$f + 12$

7.3 Duality in Linear Programming

Associated with every LP problem is another problem called the *dual*. The original LP is called the primal problem. Some theorems related to dual and primal problems are stated and explained. Dual variables are related to Lagrange multipliers of the primal constraints. Solution of the dual problem can be recovered from the final primal solution, and vice versa. Therefore, only one of the two problems needs to be solved. This is illustrated with examples.

7.3.1 Standard Primal LP

There are several ways of defining the primal and the corresponding dual problems. We shall define a *standard primal problem* as: find x_1, x_2, \dots, x_n to maximize a primal objective function

$$z_p = d_1x_1 + \dots + d_nx_n = \sum_{i=1}^n d_i x_i = \mathbf{d}^T \mathbf{x} \quad (7.8)$$

subject to the constraints

$$\begin{aligned} a_{11}x_1 + \dots + a_{1n}x_n &\leq e_1 \\ \dots & \\ a_{m1}x_1 + \dots + a_{mn}x_n &\leq e_m \\ x_j &\geq 0; \quad j = 1 \text{ to } n \end{aligned} \quad (\mathbf{Ax} \leq \mathbf{e}) \quad (7.9)$$

We shall use a subscript p on z to indicate the primal objective function. Also z is used as the maximization function. It must be understood that in the standard LP problem defined in Eqs. (6.4) to (6.6), all constraints were equalities and right side parameters b_i were non-negative. However, in the definition of the standard primal problem, all constraints must be “ \leq type” and there is no restriction on the sign of the right side parameters e_i . So, “ \geq type” constraints must be multiplied by -1 to convert them to “ \leq type.” Equalities should be also converted to “ \leq type” constraints. This is explained later in this section. Note that to solve the preceding primal LP problem by the Simplex method, we must transform it into the standard Simplex form of Eqs. (6.4) to (6.6).

7.3.2 Dual LP Problem

The dual for the standard primal is defined as follows: find dual variables y_1, y_2, \dots, y_m to minimize a dual objective function

$$f_d = e_1 y_1 + \dots + e_m y_m = \sum_{i=1}^m e_i y_i = \mathbf{e}^T \mathbf{y} \quad (7.10)$$

subject to the constraints

$$\begin{aligned} a_{11} y_1 + \dots + a_{m1} y_m &\geq d_1 \\ \dots & \\ a_{1n} y_1 + \dots + a_{mn} y_m &\geq d_n \\ y_i &\geq 0; \quad i = 1 \text{ to } m \end{aligned} \quad (\mathbf{A}^T \mathbf{y} \geq \mathbf{d}) \quad (7.11)$$

We use a subscript d on f to indicate that it is the cost function for the dual problem. Note the following *relations between the primal and dual problems*:

1. The number of dual variables is equal to the number of primal constraints. Each dual variable is associated with a primal constraint. For example, y_i is associated with the i th primal constraint.
2. The number of dual constraints is equal to the number of primal variables. Each primal variable is associated with a dual constraint. For example, x_i is associated with the i th dual constraint.
3. Primal constraints are “ \leq type” inequalities, whereas the dual constraints are “ \geq type.”
4. The maximization of the primal objective function is replaced by the minimization of the dual cost function.
5. The coefficients d_i of the primal objective function become the right side of the dual constraints. The right side parameters e_i of the primal constraints become coefficients for the dual cost function.
6. The coefficient matrix $[a_{ij}]$ of the primal constraints is transposed to $[a_{ji}]$ for dual constraints.
7. The nonnegativity condition applies to both primal and dual variables.

Example 7.2 illustrates how to write the dual problem for a given LP problem.

EXAMPLE 7.2 Dual of an LP Problem

Write the dual of the problem: maximize $z_p = 5x_1 - 2x_2$ subject to $2x_1 + x_2 \leq 9$, $x_1 - 2x_2 \leq 2$, $-3x_1 + 2x_2 \leq 3$, $x_1, x_2 \geq 0$.

Solution. The problem is already in the standard primal form and the following associated vectors and matrices can be identified:

$$\mathbf{d} = \begin{bmatrix} 5 \\ -2 \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} 9 \\ 2 \\ 3 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & -2 \\ -3 & 2 \end{bmatrix} \quad (\text{a})$$

Since there are three primal constraints, there are three dual variables for the problem. Let y_1 , y_2 , and y_3 be the dual variables associated with the three constraints. Therefore, Eqs. (7.10) and (7.11) give the dual for the problem as: minimize $f_d = 9y_1 + 2y_2 + 3y_3$ subject to $2y_1 + y_2 - 3y_3 \geq 5$, $y_1 - 2y_2 + 2y_3 \geq -2$, $y_1, y_2, y_3 \geq 0$.

7.3.3 Treatment of Equality Constraints

Many design problems have equality constraints. Each equality constraint can be replaced by a pair of inequalities. For example, $2x_1 + 3x_2 = 5$ can be replaced by the pair $2x_1 + 3x_2 \geq 5$ and $2x_1 + 3x_2 \leq 5$. We can multiply the “ \geq type” inequality by -1 to convert it into the standard primal form. Example 7.3 illustrates treatment of equality and “ \geq type” constraints.

EXAMPLE 7.3 Dual of an LP with Equality and “ \geq Type” Constraints

Write the dual for the problem: maximize $z_p = x_1 + 4x_2$ subject to $x_1 + 2x_2 \leq 5$, $2x_1 + x_2 = 4$, $x_1 - x_2 \geq 1$, $x_1, x_2 \geq 0$.

Solution. The equality constraint $2x_1 + x_2 = 4$ is equivalent to the two inequalities $2x_1 + x_2 \geq 4$ and $2x_1 + x_2 \leq 4$. The “ \geq type” constraints are multiplied by -1 to convert them into the “ \leq ” form. Thus, the standard primal of the above problem is:

$$\text{maximize } z_p = x_1 + 4x_2 \quad (\text{a})$$

subject to

$$x_1 + 2x_2 \leq 5 \quad (\text{b})$$

$$2x_1 + x_2 \leq 4 \quad (\text{c})$$

$$-2x_1 - x_2 \leq -4 \quad (\text{d})$$

$$-x_1 + x_2 \leq -1 \quad (\text{e})$$

$$x_1, x_2 \geq 0 \quad (f)$$

Using Eqs. (7.10) and (7.11), the dual for the primal is:

$$\text{minimize } f_d = 5y_1 + 4(y_2 - y_3) - y_4 \quad (g)$$

subject to

$$y_1 + 2(y_2 - y_3) - y_4 \geq 1 \quad (h)$$

$$2y_1 + (y_2 - y_3) + y_4 \geq 4 \quad (i)$$

$$y_1, y_2, y_3, y_4 \geq 0 \quad (j)$$

7.3.4 Alternate Treatment of Equality Constraints

We will show that it is not necessary to replace an equality constraint by a pair of inequalities to write the dual. Note that there are four dual variables for Example 7.3. The variables y_2 and y_3 correspond to the second and third primal constraints written in the standard form. The second and third constraints are actually equivalent to the original equality constraint. Note also that the term $(y_2 - y_3)$ appears in all the expressions of the dual problem. We define $y_5 = y_2 - y_3$, which can be positive, negative, or zero, since it is the difference of two non-negative variables ($y_2 \geq 0, y_3 \geq 0$). Substituting for y_5 , the dual problem in Example 7.3 is rewritten as:

$$\text{minimize } f_d = 5y_1 + 4y_5 - y_4 \quad (a)$$

subject to

$$y_1 + 2y_5 - y_4 \geq 1 \quad (b)$$

$$2y_1 + y_5 + y_4 \geq 4 \quad (c)$$

$$y_1, y_4 \geq 0; y_5 = y_2 - y_3 \text{ is unrestricted in sign} \quad (d)$$

The number of dual variables is now only three. Since the number of dual variables is equal to the number of the original primal constraints, the dual variable y_5 must be associated with the equality constraint $2x_1 + x_2 = 4$. Thus, we can draw the following conclusion: *If the i th primal constraint is left as an equality, the i th dual variable is unrestricted in sign.* In a similar manner, we can show that if the primal variable is unrestricted in sign, then the i th dual constraint is an equality. This is left as an exercise. Example 7.4 demonstrates recovery of the primal formulation from the dual formulation.

EXAMPLE 7.4 Recovery of Primal Formulation from Dual Formulation

Note that we can convert a dual problem into the standard primal form and write its dual again. It can be shown that the dual of this problem gives the primal problem back again. To see this, let us convert the preceding dual problem into standard primal form:

$$\text{maximize } z_p = -5y_1 - 4y_5 + y_4 \quad (\text{a})$$

subject to

$$-y_1 - 2y_5 + y_4 \leq -1 \quad (\text{b})$$

$$-2y_1 - y_5 - y_4 \leq -4 \quad (\text{c})$$

$$y_1, y_4 \geq 0; y_5 = y_2 - y_3 \text{ is unrestricted in sign} \quad (\text{d})$$

Writing the dual of the above problem, we obtain:

$$\text{minimize } f_d = -x_1 - 4x_2 \quad (\text{e})$$

subject to

$$-x_1 - 2x_2 \geq -5 \quad (\text{f})$$

$$-2x_1 - x_2 = -4 \quad (\text{g})$$

$$x_1 - x_2 \geq 1 \quad (\text{h})$$

$$x_1, x_2 \geq 0 \quad (\text{i})$$

which is the same as the original problem (Example 7.3). Note that in the preceding dual problem, the second constraint is an equality because the second primal variable (y_5) is unrestricted in sign. Theorem 7.1 states this result.

Theorem 7.1 Dual of Dual The dual of the dual problem is the primal problem.

7.3.5 Determination of Primal Solution from Dual Solution

It remains to be determined how the optimum solution of the primal is obtained from the optimum solution of the dual or vice versa. First, let us multiply each inequality in Eq. (7.11) by x_1, x_2, \dots, x_n and add them. Since x_j 's are restricted to be nonnegative, we get the inequality

$$x_1(a_{11}y_1 + \dots + a_{m1}y_m) + x_2(a_{12}y_1 + \dots + a_{m2}y_m) + \dots + x_n(a_{1n}y_1 + \dots + a_{mn}y_m) \geq d_1x_1 + d_2x_2 + \dots + d_nx_n \quad (7.12a)$$

In the matrix form, the above inequality is written as $\mathbf{x}^T \mathbf{A}^T \mathbf{y} \geq \mathbf{x}^T \mathbf{d}$. Rearranging the equation by collecting terms with y_1, y_2, \dots, y_m (or taking transpose of the left side as $\mathbf{y}^T \mathbf{A} \mathbf{x}$), we obtain

$$y_1(a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n) + y_2(a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n) + \dots + y_m(a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n) \geq d_1x_1 + d_2x_2 + \dots + d_nx_n \quad (7.12b)$$

In the matrix form the above inequality can be written as $\mathbf{y}^T \mathbf{A} \mathbf{x} \geq \mathbf{x}^T \mathbf{d}$. Each term in parentheses in Eq. (7.12b) is less than the corresponding value of e on the right side of Inequalities (7.9). Therefore, replacing these terms with the corresponding e 's from Inequalities (7.9) preserves the inequality in Eq. (7.12b).

$$y_1e_1 + y_2e_2 + \dots + y_me_m \geq d_1x_1 + d_2x_2 + \dots + d_nx_n, \text{ or } \mathbf{y}^T \mathbf{e} \geq \mathbf{x}^T \mathbf{d} \quad (7.13)$$

Note that in Inequality (7.13) the left side is the dual cost function and the right side is the primal objective function. Therefore, from Inequality (7.13), $f_d \geq z_p$ for all (x_1, x_2, \dots, x_n) and (y_1, y_2, \dots, y_m) satisfying Eqs. (7.8) to (7.11). Thus, the vectors \mathbf{x} and \mathbf{y} with $z_p = f_d$, maximize z_p while minimizing f_d . The optimum (minimum) value of the dual cost function is also the optimum (maximum) value of the primal objective function. Theorems 7.2, 7.3, and 7.4 regarding primal and dual problems can be stated as follows.

Theorem 7.2 Relationship Between Primal and Dual Let \mathbf{x} and \mathbf{y} be in the feasible sets of primal and dual problems, respectively [defined in Eqs. (7.8) to (7.11)]. Then the following conditions hold:

1. $f_d(\mathbf{y}) \geq z_p(\mathbf{x})$.
2. If $f_d = z_p$, then \mathbf{x} and \mathbf{y} are solutions for the primal and dual problems, respectively.
3. If the primal is unbounded, the corresponding dual is infeasible, and vice versa.
4. If the primal is feasible and the dual is infeasible, then the primal is unbounded and vice versa.

Theorem 7.3 Primal and Dual Solutions Let both the primal and dual have feasible points. Then both have optimum solution in \mathbf{x} and \mathbf{y} respectively, and $f_d(\mathbf{y}) = z_p(\mathbf{x})$.

Theorem 7.4 Solution of Primal from Dual If the i th dual constraint is a strict inequality at optimum, then the corresponding i th primal variable is nonbasic, i.e. it vanishes. Also, if the i th dual variable is basic, then the i th primal constraint is satisfied at equality.

The conditions of Theorem 7.4 can be written as

$$\text{if } \sum_{i=1}^m a_{ij}y_i > d_j, \text{ then } x_j = 0 \quad (a)$$

(if the j th dual constraint is a strict inequality, then the j th primal variable is nonbasic)

$$\text{if } y_i > 0, \text{ then } \sum_{j=1}^n a_{ij}x_j = e_i \quad (b)$$

(if the i th dual variable is basic, the i th primal constraint is an equality).

These conditions can be used to obtain primal variables using the dual variables. The primal constraints satisfied at equality are identified from values of the dual variables. The resulting linear equations can be solved simultaneously for the primal variables. However, this is not necessary as the final dual tableau can be used directly to obtain primal variables. We shall illustrate the use of these theorems in Example 7.5.

EXAMPLE 7.5 Primal and Dual Solutions

Consider the following problem:

$$\text{maximize } z_p = 5x_1 - 2x_2 \quad (\text{a})$$

subject to

$$2x_1 + x_2 \leq 9 \quad (\text{b})$$

$$x_1 - 2x_2 \leq 2 \quad (\text{c})$$

$$-3x_1 + 2x_2 \leq 3 \quad (\text{d})$$

$$x_1, x_2 \geq 0 \quad (\text{e})$$

Solve the primal and the dual problems and study their final tableaux.

Solution. The problem has been solved using the Simplex method in Example 6.15 and Table 6-18. The final tableau is reproduced from there in Table 7-2. From the final primal tableau,

$$\begin{array}{ll} \text{basic variables:} & x_1 = 4, x_2 = 1, x_5 = 13 \\ \text{nonbasic variables:} & x_3 = 0, x_4 = 0 \\ \text{maximum objective function:} & z_p = 18 \text{ (minimum value is } -18) \end{array}$$

Now, let us write the dual for the problem and solve it using the Simplex method. Note that the original problem is already in the standard primal form. There are three primal inequality constraints so there are three dual variables. There are two primal variables so there are two dual constraints. Let y_1 , y_2 , and y_3 be the dual variables. Therefore, the dual of the problem is given as:

$$\text{minimize } f_d = 9y_1 + 2y_2 + 3y_3 \quad (\text{f})$$

subject to

$$2y_1 + y_2 - 3y_3 \geq 5 \quad (\text{g})$$

TABLE 7-2 Final Tableau for Example 7.5 by Simplex Method (Primal Solution)

Basic ↓	x_1	x_2	x_3	x_4	x_5	b
x_2	0	1	0.2	-0.4	0	1
x_1	1	0	0.4	0.2	0	4
x_5	0	0	0.8	1.4	1	13
Cost	0	0	1.6	1.8	0	$f_p + 18$

$$y_1 - 2y_2 + 2y_3 \geq -2 \quad (\text{h})$$

$$y_1, y_2, y_3 \geq 0 \quad (\text{i})$$

Writing the constraints in the standard Simplex form by introducing slack, surplus, and artificial variables, we obtain

$$2y_1 + y_2 - 3y_3 - y_4 + y_6 = 5 \quad (\text{j})$$

$$y_1 - 2y_2 - 2y_3 + y_5 = 2 \quad (\text{k})$$

$$y_i \geq 0; \quad i = 1 \text{ to } 6 \quad (\text{l})$$

where y_4 is a surplus variable, y_5 is a slack variable, and y_6 is an artificial variable. The two-phase Simplex procedure can be used to solve the problem. Thus, we obtain the sequence of calculations for the dual problem shown in Table 7-3. From the final dual tableau, we obtain the following solution:

$$\begin{array}{ll} \text{basic variables:} & y_1 = 1.6, y_2 = 1.8 \\ \text{nonbasic variables:} & y_3 = 0, y_4 = 0, y_5 = 0 \\ \text{minimum value of dual function:} & f_d = 18 \end{array}$$

TABLE 7-3 Solution of Dual of the Problem in Example 7.5

Initial tableau: y_6 is identified to be replaced with y_1 in the basic set.							
<i>Basic</i> ↓	y_1	y_2	y_3	y_4	y_5	y_6	b
y_6	2	1	-3	-1	0	1	5
y_5	-1	2	-2	0	1	0	2
Cost	9	2	3	0	0	0	$f_d - 0$
Artificial cost	-2	-1	3	1	0	0	$w - 5$
Second tableau: End of Phase I. y_5 is identified to be replaced with y_2 in the basic set.							
<i>Basic</i> ↓	y_1	y_2	y_3	y_4	y_5	y_6	b
y_1	1	0.5	-1.5	-0.5	0	0.5	2.5
y_5	0	2.5	-3.5	-0.5	1	0.5	4.5
Cost	0	-2.5	16.5	4.5	0	-4.5	$f_d - 22.5$
Artificial cost	0	0	0	0	0	1	$w - 0$
Third tableau: Reduced cost coefficients in nonbasic columns are nonnegative; the tableau gives optimum point. End of Phase II.							
<i>Basic</i> ↓	y_1	y_2	y_3	y_4	y_5	y_6	b
y_1	1	0	-0.8	-0.4	-0.2	0.4	1.6
y_2	0	1	-1.4	-0.2	0.4	0.2	1.8
Cost	0	0	13.0	4.0	1.0	-4.0	$f_d - 18$

Note that at the optimum $f_d = z_p$, which satisfies the conditions of Theorems 7.2 and 7.3. Using Theorem 7.4, we see that the first and second primal constraints must be satisfied at equality since the dual variables y_1 and y_2 associated with the constraints are positive (basic) in Table 7-3. Therefore, primal variables x_1 and x_2 are obtained as a solution of the first two primal constraints satisfied at equality: $2x_1 + x_2 = 9$, $x_1 - 2x_2 = 2$. The solution of these equations is given as $x_1 = 4$, $x_2 = 1$ which is the same as obtained from the final primal tableau.

7.3.6 Use of Dual Tableau to Recover Primal Solution

It turns out that we do not need to follow the preceding procedure (use of Theorem 7.4) to recover the primal variables. The final dual tableau contains all the information to recover the primal solution. Similarly, the final primal tableau contains all the information to recover the dual solution. Looking at the final tableau in Table 7-3 for Example 7.5, we observe that the elements in the last row of the dual tableau match the elements in the last column of the primal tableau in Table 7-2. Similarly, the reduced cost coefficients in the final primal tableau match the dual variables. To recover the primal variables from the final dual tableau, we use reduced cost coefficients in columns corresponding to slack or surplus variables. We note that the reduced cost coefficient in column y_4 is precisely x_1 and the reduced cost coefficient in column y_5 is precisely x_2 . Therefore, reduced cost coefficients corresponding to slack and surplus variables in the final dual tableau give values of primal variables. Similarly, if we solve the primal problem, we can recover the dual solution from the final primal tableau. Theorem 7.5 summarizes this result.

Theorem 7.5 Recovery of Primal Solution from Dual Tableau Let the dual of the standard primal defined in Eqs. (7.8) and (7.9) (i.e., maximize $\mathbf{d}^T \mathbf{x}$ subject to $\mathbf{Ax} \leq \mathbf{e}$, $\mathbf{x} \geq \mathbf{0}$) be solved by the standard Simplex method. Then the value of the i th primal variable equals the reduced cost coefficient of the slack or surplus variable associated with the i th dual constraint in the final dual tableau. In addition, if a dual variable is nonbasic, then its reduced cost coefficient equals the value of slack or surplus variable for the corresponding primal constraint.

Note that if a dual variable is nonbasic (i.e., has zero value), then its reduced cost coefficient equals the value of the slack or surplus variable for the corresponding primal constraint. In Example 7.5, y_3 , the dual variable corresponding to the third primal constraint, is nonbasic. The reduced cost coefficient in the y_3 column is 13. Therefore, the slack variable for the third primal constraint has a value of 13. This is the same as obtained from the final primal tableau. We also note that the dual solution can be obtained from the final primal tableau using Theorem 7.5 as $y_1 = 1.6$, $y_2 = 1.8$, $y_3 = 0$, which is the same solution as before. While using Theorem 7.5, the following additional points should be noted:

1. When the final primal tableau is used to recover the dual solution, the dual variables correspond to the primal constraints expressed in the “ \leq ” form only. However, the primal constraints must be converted to standard Simplex form while solving the problem. Recall that all the right sides of constraints must be nonnegative for the Simplex method. The dual variables are nonnegative only for the constraints written in the “ \leq ” form.
2. When a primal constraint is an equality, it is treated in the Simplex method by adding an artificial variable in Phase I. There is no slack or surplus variable associated with an equality. We also know from the previous discussion that the dual variable associated with the equality constraint is unrestricted in sign. Then the

question is how to recover the dual variable for the equality constraint from the final primal tableau? There are a couple of ways of doing this. The first procedure is to convert the equality constraint into a pair of inequalities, as noted previously. For example, the constraint $2x_1 + x_2 = 4$ is written as the pair of inequalities $2x_1 + x_2 \leq 4$, $-2x_1 - x_2 \leq -4$. The two inequalities are treated in a standard way in the Simplex method. The corresponding dual variables are recovered from the final primal tableau using Theorem 7.5. Let $y_2 \geq 0$ and $y_3 \geq 0$ be the dual variables associated with the two inequality constraints, respectively, and y_1 be the dual variable associated with the original equality constraint. Then, $y_1 = y_2 - y_3$. Accordingly, y_1 is unrestricted in sign and its value is known using y_2 and y_3 .

The second way of recovering the dual variable for the equality constraint is to carry along its artificial variable column in Phase II of the Simplex method. Then the dual variable for the constraint is the reduced cost coefficient in the artificial variable column in the final primal tableau. We illustrate these procedures in Example 7.6.

EXAMPLE 7.6 Use of Final Primal Tableau to Recover Dual Solutions

Solve the following LP problem and recover its dual solution from the final primal tableau:

$$\text{maximize } z_p = x_1 + 4x_2 \quad (\text{a})$$

subject to

$$x_1 + 2x_2 \leq 5, \quad (\text{b})$$

$$2x_1 + x_2 = 4, \quad (\text{c})$$

$$x_1 - x_2 \geq 1, \quad (\text{d})$$

$$x_1, x_2 \geq 0 \quad (\text{e})$$

Solution. When the equality constraint is converted into a pair of inequalities $2x_1 + x_2 \leq 4$, $-2x_1 - x_2 \leq -4$, the problem is the same as the one solved in Example 6.14. The final tableau for the problem is given in Table 6-17. Using Theorem 7.5, the dual variables for the preceding four constraints are

1. $x_1 + 2x_2 \leq 5$: $y_1 = 0$, reduced cost coefficient of x_3 , the slack variable
2. $2x_1 + x_2 \leq 4$: $y_2 = \frac{5}{3}$, reduced cost coefficient of x_4 , the slack variable
3. $-2x_1 - x_2 \leq -4$: $y_3 = 0$, reduced cost coefficient of x_5 , the surplus variable
4. $-x_1 + x_2 \leq -1$: $y_4 = \frac{7}{3}$, reduced cost coefficient of x_6 , the surplus variable

Thus, from the above discussion, the dual variable for the equality constraint $2x_1 + x_2 = 4$ is $y_2 - y_3 = \frac{5}{3}$. Note also that $y_4 = \frac{7}{3}$ is the dual variable for the fourth constraint written as $-x_1 + x_2 \leq -1$ and not for the constraint $x_1 - x_2 \geq 1$.

Now, let us re-solve the same problem with the equality constraint as it is. The problem is the same as the one solved in Example 6.16. The final tableau for the problem is given in Table 6-19. Using Theorem 7.5 and the preceding discussion, the dual variables for the given three constraints are

1. $x_1 + 2x_2 \leq 5$: $y_1 = 0$, reduced cost coefficient of x_3 , the slack variable
2. $2x_1 + x_2 = 4$: $y_2 = \frac{5}{3}$, reduced cost coefficient of x_5 , the artificial variable
3. $-x_1 + x_2 \leq -1$: $y_3 = \frac{7}{3}$, reduced cost coefficient of x_4 , the surplus variable

We see that the two solutions are the same. Therefore, we do not have to replace an equality constraint by two inequalities in the standard Simplex method. The reduced cost coefficient corresponding to the artificial variable associated with the equality constraint gives the value of the dual variable for the constraint.

7.3.7 Dual Variables as Lagrange Multipliers

Section 6.5 describes how the optimum value of the cost function for the problem changes if we change right side parameters of constraints, b_i 's, the resource limits. The constraint variation sensitivity theorem 4.7 of Section 4.5 is used to study this effect. Use of that theorem requires knowledge of the Lagrange multipliers for the constraints that must be determined. It turns out that the dual variables of the problem are related to the Lagrange multipliers. Theorem 7.6 gives this relationship.

Theorem 7.6 Dual Variables as Lagrange Multipliers Let \mathbf{x} and \mathbf{y} be optimal solutions for the primal and dual problems stated in Eqs. (7.8) to (7.11), respectively. Then the dual variables \mathbf{y} are also Lagrange multipliers for primal constraints of Eq. (7.9).

Proof The theorem can be proved by writing the KKT necessary conditions of Theorem 4.6 for the primal problem defined in Eqs. (7.8) and (7.9). To write these conditions, convert the primal problem to a minimization problem and define a Lagrange function of Eq. (4.46a) as

$$\begin{aligned} L &= -\sum_{j=1}^n d_j x_j + \sum_{i=1}^m y_i \left(\sum_{j=1}^n a_{ij} x_j - e_i \right) - \sum_{j=1}^n v_j x_j \\ &= -\mathbf{d}^T \mathbf{x} + \mathbf{y}^T (\mathbf{A}\mathbf{x} - \mathbf{e}) - \mathbf{v}^T \mathbf{x} \end{aligned} \quad (\text{a})$$

where y_i is the Lagrange multiplier for the i th primal constraint of Eq. (7.9) and v_j is the Lagrange multiplier for the j th nonnegativity constraint for the variable x_j . Write the KKT necessary conditions of Theorem 4.6 as

$$-d_j + \sum_{i=1}^m y_i a_{ij} x_j - v_j = 0; \quad j = 1 \text{ to } n \quad (\partial L / \partial x_j = 0) \quad (\text{b})$$

$$y_i \left(\sum_{j=1}^n a_{ij} x_j - e_i \right) = 0, \quad i = 1 \text{ to } m \quad (\text{c})$$

$$v_i x_i = 0, \quad x_i \geq 0, \quad i = 1 \text{ to } n \quad (\text{d})$$

$$y_i \geq 0, \quad i = 1 \text{ to } m \quad (\text{e})$$

$$v_i \geq 0, \quad i = 1 \text{ to } n \quad (\text{f})$$

Rewrite Eq. (b) as

$$-d_j + \sum_{i=1}^m a_{ij}y_i = v_j; \quad j = 1 \text{ to } n \quad (-\mathbf{d} + \mathbf{A}^T \mathbf{y} = \mathbf{v})$$

Using conditions (f) in the preceding equation, we conclude

$$\sum_{i=1}^m a_{ij}y_i \geq d_j; \quad j = 1 \text{ to } n \quad (\mathbf{A}^T \mathbf{y} \geq \mathbf{d}) \quad (\text{g})$$

Thus y_i 's are feasible solutions for the dual constraints of Eq. (7.11).

Now let x_i represent the optimum solution for the primal problem. Then, m of the x_i 's are positive (barring degeneracy), and the corresponding v_i are equal to zero from Eq. (d). The remaining x_i 's are zero and the corresponding v_i 's are greater than zero. Therefore, from Eq. (g), we obtain

$$(i) \quad v_j > 0, x_j = 0, \quad \sum_{i=1}^m a_{ij}y_i > d_j \quad (\text{h})$$

$$(ii) \quad v_j = 0, x_j > 0, \quad \sum_{i=1}^m a_{ij}y_i = d_j \quad (\text{i})$$

Now adding the m rows given in Eq. (c), interchanging the sums on the left side, and rearranging, we obtain

$$\sum_{j=1}^n x_j \sum_{i=1}^m a_{ij}y_i = \sum_{i=1}^m y_i e_i \quad (\mathbf{x}^T \mathbf{A}^T \mathbf{y} = \mathbf{y}^T \mathbf{e}) \quad (\text{j})$$

Using Eqs. (h) and (i), Eq. (j) can be written as

$$\sum_{j=1}^n d_j x_j = \sum_{i=1}^m y_i e_i \quad (\mathbf{d}^T \mathbf{x} = \mathbf{y}^T \mathbf{e}) \quad (\text{k})$$

Equation (k) also states that

$$z_p = \sum_{i=1}^m y_i e_i = \mathbf{y}^T \mathbf{e} \quad (\text{l})$$

The right side of Eq. (l) represents the dual cost function. According to Theorem 7.2, if the primal and dual functions have the same values and if \mathbf{x} and \mathbf{y} are feasible points for the primal and the dual problems, then they are optimum solutions for the respective problems. Thus, the Lagrange multipliers y_i , $i = 1$ to m solve the dual problem defined in Eqs. (7.10) and (7.11).

Exercises for Chapter 7

Write dual problem for the following problems; solve the dual problems and recover the values of the primal variables from the final dual tableau; verify the solution graphically whenever possible.

- | | | | | | |
|------|---------------|------|---------------|------|---------------|
| 7.1 | Exercise 6.55 | 7.2 | Exercise 6.56 | 7.3 | Exercise 6.57 |
| 7.4 | Exercise 6.58 | 7.5 | Exercise 6.59 | 7.6 | Exercise 6.60 |
| 7.7 | Exercise 6.61 | 7.8 | Exercise 6.62 | 7.9 | Exercise 6.63 |
| 7.10 | Exercise 6.64 | 7.11 | Exercise 6.65 | 7.12 | Exercise 6.66 |
| 7.13 | Exercise 6.67 | 7.14 | Exercise 6.68 | 7.15 | Exercise 6.69 |
| 7.16 | Exercise 6.70 | 7.17 | Exercise 6.71 | 7.18 | Exercise 6.72 |
| 7.19 | Exercise 6.73 | 7.20 | Exercise 6.74 | 7.21 | Exercise 6.75 |
| 7.22 | Exercise 6.76 | | | | |

8 Numerical Methods for Unconstrained Optimum Design

Upon completion of this chapter, you will be able to:

- Explain the concept of iterative numerical search methods for optimum design
- Verify the descent condition for a given search direction
- Explain two basic calculations in the numerical search methods for optimum design: (1) calculation of a search direction and (2) calculation of a step size in the search direction
- Calculate the search direction for the steepest descent and conjugate gradient methods
- Calculate a step size along the search direction using the golden sections method

When some or all of the functions of the problem (cost function and/or constraint functions) are nonlinear for an optimization problem, it is called a *nonlinear programming* (NLP) problem. This chapter and the next chapter concentrate on the concepts and description of methods for unconstrained nonlinear optimization problems. Chapters 10 and 11 treat constrained problems.

Numerical methods for nonlinear optimization problems are needed because the analytical methods for solving some of the problems are too cumbersome to use. With the analytical methods described in Chapters 4 and 5, we write the necessary conditions of optimality and solve them for candidate local minimum designs. There are two basic reasons why the methods are inappropriate for many engineering design problems:

1. *The numbers of design variables and constraints can be large.* In that case, the necessary conditions give a large number of nonlinear equations, which can be difficult to solve. Numerical methods must be used to find solutions of such equations in any case. Therefore it is appropriate to use the numerical methods directly to solve the optimization problems. Even if the problem is not large, these equations can be highly nonlinear and cannot be solved in a closed form.
2. In many engineering applications, *cost and/or constraint functions are implicit functions of the design variables*; that is, explicit functional forms in terms of the

independent variables are not known. These functions cannot be treated easily in the analytical methods for solution of optimality conditions.

For these reasons, we must develop systematic numerical approaches for the optimum design of engineering systems. In such approaches, we estimate an initial design and improve it until optimality conditions are satisfied. Many *numerical methods* have been developed for NLP problems. Some are better than others and research in the area continues to develop still better techniques. Detailed derivations and theory of various methods are beyond the scope of the present text. However, it is important to understand a few basic concepts, ideas, and procedures that are used in most algorithms for unconstrained and constrained optimization. Therefore, *the approach followed in this text is to stress these underlying concepts with example problems*. Some details of the numerical algorithms are also presented to give the student a flavor of the calculations performed in search for optimum solutions.

In the present chapter, unconstrained optimization problems are treated. Numerical details for some algorithms are presented and discussed to show the type of calculations needed to solve nonlinear optimization problems. These algorithms are rarely done “by hand”; they require a computer program for their effective use. *Most information technology centers either have or can easily acquire some standard computer programs for general use, such as MATLAB, Excel, and so on. Therefore, coding of the algorithms should be attempted only as a last resort.* It is, however, important to understand the underlying ideas to use the methods properly. In addition, many of the concepts are applicable to constrained optimization problems as well.

Some numerical problems are solved using the programs given in Appendix D to study the performance of the algorithms and compare them. It must be understood that the behavior of any algorithm can be drastically affected by the numerical implementation details. In addition, *numerical performance* of a program can change from one computer system to another and from one compiler to another.

The unconstrained optimization problems are classified as one-dimensional and multidimensional problems as shown in Fig. 8-1. Numerical methods for solving unconstrained problems have been developed over the last several decades. Substantial work, however, was done during the 1950s and 1960s because it was shown that constrained optimization problems could be transformed to a sequence of unconstrained problems (procedures for transformations are presented in Chapter 9). Consequently, the methods have gained considerable importance, and substantial effort has been expended in developing efficient algorithms and computer programs for unconstrained optimization problems.

8.1 General Concepts Related to Numerical Algorithms

In this section, we describe some basic concepts that are applicable to both constrained and unconstrained numerical optimization methods. The idea of iterative numerical algorithms is

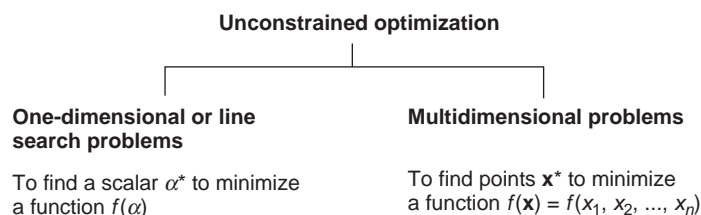


FIGURE 8-1 Classification of unconstrained optimization problems.

introduced to search for optimum solutions for the design problem. The algorithms are initiated with an estimate for the optimum solution that is improved iteratively if it does not satisfy the optimality conditions. Since we want to minimize the cost function, the idea of a descent step is introduced, which simply means that changes in the design at every search step must reduce the cost function value. Convergence of an algorithm and its rate of convergence are also briefly described.

8.1.1 A General Algorithm

Many numerical solution methods are described by the following iterative prescription:

Vector form:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}; \quad k = 0, 1, 2, \dots \quad (8.1)$$

Component form:

$$x_i^{(k+1)} = x_i^{(k)} + \Delta x_i^{(k)}; \quad i = 1 \text{ to } n; \quad k = 0, 1, 2, \dots \quad (8.2)$$

In these equations, the superscript k represents the iteration number, subscript i denotes the design variable number, $\mathbf{x}^{(0)}$ is a starting point, and $\Delta\mathbf{x}^{(k)}$ is a change in the current point. The iterative scheme described in Eq. (8.1) or (8.2) is continued until optimality conditions are satisfied or an acceptable solution is obtained. The iterative formula is applicable to constrained as well as unconstrained problems. For unconstrained problems, calculations for $\Delta\mathbf{x}^{(k)}$ depend on the cost function and its derivatives at the current design point. For constrained problems, the constraints must also be considered while computing the change in design $\Delta\mathbf{x}^{(k)}$. Therefore, in addition to the cost function and its derivatives, the constraint functions and their derivatives play a role in determining $\Delta\mathbf{x}^{(k)}$. There are several methods for calculating $\Delta\mathbf{x}^{(k)}$ for unconstrained and constrained problems. We shall describe some of the basic methods for unconstrained problems later in this chapter.

The change in design $\Delta\mathbf{x}^{(k)}$ is further decomposed into two parts as

$$\Delta\mathbf{x}^{(k)} = \alpha_k \mathbf{d}^{(k)} \quad (8.3)$$

where $\mathbf{d}^{(k)}$ is a “desirable” search direction in the design space and α_k is a positive scalar called the step size in that direction. If the direction $\mathbf{d}^{(k)}$ is any “good,” then the step size must be greater than 0; this shall become clearer when we relate the search direction to a descent direction for the cost function. Thus, the process of computing $\Delta\mathbf{x}^{(k)}$ involves solving two separate subproblems: the direction finding subproblem and the step length determination subproblem (scaling along the direction). The process of moving from one design point to the next is illustrated in Fig. 8-2. In the figure, B is the current design point, $\mathbf{d}^{(k)}$ is the search direction, and α_k is a step length. Therefore, when $\alpha_k \mathbf{d}^{(k)}$ is added to the current design, we reach a new point C in the design space. The entire process is repeated from point C. There are many procedures for calculating the step size α_k and the search direction $\mathbf{d}^{(k)}$. Various combinations of the procedures can be used to develop different optimization algorithms.

In summary, the *basic idea of numerical methods* for nonlinear optimization problems is to start with a reasonable estimate for the optimum design. Cost and constraint functions and their derivatives are evaluated at that point. Based on them, the design is moved to a new point. The process is continued until either optimality conditions or some other stopping criteria are met. This iterative process represents an organized search through the design space for points that represent local minima. Thus, the procedures are often called the *search techniques* or *direct methods* of optimization. The iterative process is summarized as a *general algorithm* that is applicable to both constrained and unconstrained problems:

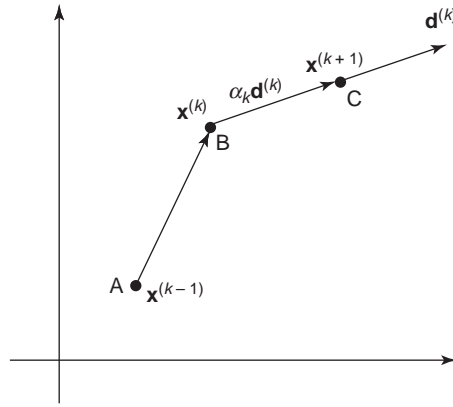


FIGURE 8-2 Conceptual diagram for iterative steps of an optimization method.

Step 1. Estimate a reasonable starting design $\mathbf{x}^{(0)}$. Set the iteration counter $k = 0$.

Step 2. Compute a search direction $\mathbf{d}^{(k)}$ in the design space. This calculation generally requires a cost function value and its gradient for unconstrained problems and, in addition, constraint functions and their gradients for constrained problems.

Step 3. Check for convergence of the algorithm. If it has converged, stop; otherwise, continue.

Step 4. Calculate a positive step size α_k in the direction $\mathbf{d}^{(k)}$.

Step 5. Update the design as follows, set $k = k + 1$ and go to Step 2:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)} \quad (8.4)$$

In the remaining sections of this chapter, we shall present some methods for calculating the step size α_k and the search direction $\mathbf{d}^{(k)}$ for unconstrained optimization problems.

8.1.2 Descent Direction and Descent Step

We have referred to $\mathbf{d}^{(k)}$ as a desirable direction of design change in the iterative process. Now we discuss what we mean by a *desirable direction*. The objective of the iterative optimization process is to reach a minimum point for the cost function $f(\mathbf{x})$. Let us assume that we are in the k th iteration and we have determined that $\mathbf{x}^{(k)}$ is not a minimum point, i.e., the optimality conditions of Theorem 4.4 are not satisfied. If $\mathbf{x}^{(k)}$ is not a minimum point, then we should be able to find another point $\mathbf{x}^{(k+1)}$ with a smaller cost function value than the one at $\mathbf{x}^{(k)}$. This statement can be expressed mathematically as

$$f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)}) \quad (8.5)$$

Substitute $\mathbf{x}^{(k+1)}$ from Eq. (8.4) into the preceding inequality to obtain

$$f(\mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)}) \quad (8.6)$$

Approximating the left side of Eq. (8.6) by the linear Taylor's expansion at the point $\mathbf{x}^{(k)}$, we get

$$f(\mathbf{x}^{(k)}) + \alpha_k (\mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)}) \quad (8.7)$$

where $\mathbf{c}^{(k)} = \nabla f(\mathbf{x}^{(k)})$ is the gradient of $f(\mathbf{x})$ at the point $\mathbf{x}^{(k)}$ and (\cdot) represents the dot product of the two vectors. Subtracting $f(\mathbf{x}^{(k)})$ from both sides of Inequality (8.7), we get $\alpha_k(\mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)}) < 0$. Since $\alpha_k > 0$, it may be dropped without affecting the inequality. Therefore, we get the condition

$$\mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)} < 0 \quad (8.8)$$

Since $\mathbf{c}^{(k)}$ is a known quantity (gradient of the cost function), the search direction $\mathbf{d}^{(k)}$ must be calculated to satisfy Inequality (8.8). Any small move in such a direction will decrease the cost function. Geometrically, using the definition of the dot product of two vectors, the inequality shows that the angle between the vectors $\mathbf{c}^{(k)}$ and $\mathbf{d}^{(k)}$ must be between 90° and 270° .

We can now define a *desirable direction of change* as any vector $\mathbf{d}^{(k)}$ satisfying Inequality (8.8). Such vectors are also called *directions of descent* for the cost function, and Inequality (8.8) is called the *descent condition*. A step of the iterative optimization method based on these directions is called a *descent step*. There can be several directions of descent at a design point and each optimization algorithm computes it differently.

The descent direction is also sometimes called the “downhill” direction. The problem of minimizing $f(\mathbf{x})$ can be considered as a problem of trying to reach the bottom of a hill from a high point. From the top, we find a downhill direction and travel along it to the lowest point. From the lowest point in the direction, we repeat the process until the bottom of the hill is reached. A method based on the idea of a descent step is called a *descent method*. Clearly, such a method will not converge to a local maximum point for the function. The concepts of *descent directions* and descent step are used in most numerical optimization methods. Therefore, they should be clearly understood. Example 8.1 illustrates the concept of a descent direction.

EXAMPLE 8.1 Check for the Descent Condition

For the function

$$f(\mathbf{x}) = x_1^2 - x_1x_2 + 2x_2^2 - 2x_1 + e^{(x_1+x_2)} \quad (a)$$

check if the direction $\mathbf{d} = (1, 2)$ at the point $(0, 0)$ is a descent direction for the function f .

Solution. If $\mathbf{d} = (1, 2)$ is a descent direction, then it must satisfy Inequality (8.8). To verify this, we calculate the gradient \mathbf{c} of the function $f(\mathbf{x})$ at $(0, 0)$ and evaluate $(\mathbf{c} \cdot \mathbf{d})$, as

$$\mathbf{c} = (2x_1 - x_2 - 2 + e^{(x_1+x_2)}, -x_1 + 4x_2 + e^{(x_1+x_2)}) = (-1, 1) \quad (b)$$

$$(\mathbf{c} \cdot \mathbf{d}) = (-1, 1) \begin{bmatrix} 1 \\ 2 \end{bmatrix} = -1 + 2 = 1 > 0 \quad (c)$$

Inequality (8.8) is violated, and thus the given \mathbf{d} is not a descent direction for the function $f(\mathbf{x})$.

8.1.3 Convergence of Algorithms

The central idea behind numerical methods of optimization is to search for the optimum point in an iterative manner, generating a sequence of designs. It is important to note that the success of a method depends on the guarantee of convergence of the sequence to the optimum point. The property of convergence to a local optimum point irrespective of the starting point is called *global convergence* of the numerical method. It is desirable to employ such convergent numerical methods in practice since they are more reliable. For unconstrained problems, a convergent algorithm must reduce the cost function at each iteration until a minimum point is reached. It is important to note that the algorithms converge to a local minimum point only, as opposed to a global minimum, since they only use the local information about the cost function and its derivatives in the search process. Methods to search for global minima are described in Chapter 18.

8.1.4 Rate of Convergence

In practice, a numerical method may take a large number of iterations to reach the optimum point. Therefore, it is important to employ methods having a faster rate of convergence. Rate of convergence of an algorithm is usually measured by the numbers of iterations and function evaluations needed to obtain an acceptable solution. *Rate of convergence is a measure of how fast the difference between the solution point and its estimates goes to zero.* Faster algorithms usually use second-order information about the problem functions when calculating the search direction. They are known as *Newton methods*. Many algorithms also approximate second-order information using only the first-order information. They are known as *quasi-Newton methods*, described in Chapter 9.

8.2 Basic Ideas and Algorithms for Step Size Determination

Unconstrained numerical optimization methods are based on the iterative formula given in Eq. (8.1). As discussed earlier, the problem of obtaining the design change $\Delta \mathbf{x}$ is usually decomposed into two subproblems: (1) direction finding and (2) step size determination, as expressed in Eq. (8.3). We need to discuss numerical methods for solving both subproblems. In the following paragraphs, we first discuss the problem of *step size determination*. This is often called the *one-dimensional search (or, line search)* problem. Such problems are simpler to solve. This is one reason for discussing them first. Following one-dimensional minimization methods, two methods are described in Sections 8.3 and 8.4 for finding a “desirable” *search direction* \mathbf{d} in the design space.

8.2.1 Definition of One-Dimensional Minimization Subproblem

For an optimization problem with several variables, the direction finding problem must be solved first. Then, a step size must be determined by searching for the minimum of the cost function along the search direction. This is always a one-dimensional minimization problem. To see how the line search will be used in multidimensional problems, let us assume for the moment that a search direction $\mathbf{d}^{(k)}$ has been found. Then, in Eqs. (8.1) and (8.3), scalar α_k is the only unknown. Since the best step size α_k is yet unknown, we replace it by α in Eq. (8.3). Then, using Eqs. (8.1) and (8.3), the cost function $f(\mathbf{x})$ is given as $f(\mathbf{x}^{(k+1)}) = f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$. Now, since $\mathbf{d}^{(k)}$ is known, the right side becomes a function of the scalar parameter α only. This process is summarized in the following equations:

Design update:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)} \quad (8.9a)$$

Cost function evaluation:

$$f(\mathbf{x}^{(k+1)}) = f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}) = \bar{f}(\alpha) \quad (8.9b)$$

where $\bar{f}(\alpha)$ is the new function with α as the only independent variable (in the sequel, we shall drop the overbar for functions of single variable). Note that at $\alpha = 0$, $f(0) = f(\mathbf{x}^{(k)})$ from Eq. (8.9b), which is the current value of the cost function. It is important to understand this *reduction of a function of n variables to a function of only one variable* since this fundamental step is used in almost all optimization methods. It is also important to understand the geometric significance of Eq. (8.9b). We shall elaborate on these ideas later.

If $\mathbf{x}^{(k)}$ is not a minimum point, then it is possible to find a descent direction $\mathbf{d}^{(k)}$ at the point and reduce the cost function further. *Recall that a small move along $\mathbf{d}^{(k)}$ reduces the cost function.* Therefore, using Eqs. (8.5) and (8.9b), the descent condition for the cost function can be expressed as the inequality:

$$f(\alpha) < f(0) \quad (8.10)$$

Since $f(\alpha)$ is a function of single variable, we can plot $f(\alpha)$ versus α . To satisfy Inequality (8.10), the curve $f(\alpha)$ versus α must have a negative slope at the point $\alpha = 0$. Such a curve is shown by the solid line in Fig. 8-3. It must be understood that if the search direction is that of descent, the graph of $f(\alpha)$ versus α cannot be the one shown by the dashed curve because any positive α would cause the function $f(\alpha)$ to increase, violating Inequality (8.10). This would also be a contradiction as $\mathbf{d}^{(k)}$ is a direction of descent for the cost function. Therefore, the graph of $f(\alpha)$ versus α must be the solid curve in Fig. 8-3 for all problems. In fact, the slope of the curve $f(\alpha)$ at $\alpha = 0$ is calculated as $f'(0) = \mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)}$, which is negative as seen in Eq. (8.8). This discussion shows that if $\mathbf{d}^{(k)}$ is a descent direction, then α *must always be a positive scalar* in Eq. (8.8). Thus, the *one-dimensional minimization problem* is to find $\alpha_k = \alpha$ such that $f(\alpha)$ is minimized.

8.2.2 Analytical Method to Compute Step Size

If $f(\alpha)$ is a simple function, then we can use the analytical procedure to determine α_k (necessary and sufficient conditions of Section 4.3). The necessary condition is $df(\alpha_k)/d\alpha = 0$, and the sufficient condition is $d^2f(\alpha_k)/d\alpha^2 > 0$. We shall illustrate the analytical line search

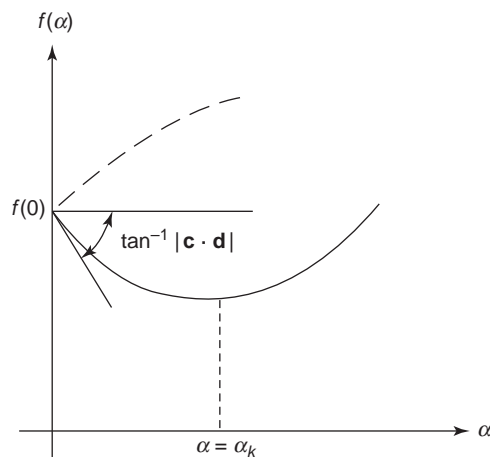


FIGURE 8-3 Graph of $f(\alpha)$ versus α .

procedure with Example 8.2. Note that differentiation of $f(\mathbf{x}^{(k+1)})$ in Eq. (8.9b) with respect to α , using the chain rule of differentiation and setting it to zero, gives

$$\frac{df(\mathbf{x}^{(k+1)})}{d\alpha} = \frac{\partial f^T(\mathbf{x}^{(k+1)})}{\partial \mathbf{x}} \frac{d(\mathbf{x}^{(k+1)})}{d\alpha} = \nabla f(\mathbf{x}^{(k+1)}) \cdot \mathbf{d}^{(k)} = \mathbf{c}^{(k+1)} \cdot \mathbf{d}^{(k)} = 0 \quad (8.11)$$

Since the dot product of two vectors is zero in Eq. (8.11), the gradient of the cost function at the new point is orthogonal to the search direction at the k th iteration, i.e., $\mathbf{c}^{(k+1)}$ is normal to $\mathbf{d}^{(k)}$. The condition in Eq. (8.11) is important for two reasons: (1) it can be used directly to obtain an equation in terms of step size α whose smallest root gives the exact step size, and (2) it can be used to check the accuracy of the step size in a numerical procedure to calculate α and thus it is called the *line search termination criterion*. Many times numerical line search methods will give an approximate or inexact value of the step size along the search direction. The line search termination criterion is useful for determining the accuracy of the step size; i.e., for checking $\mathbf{c}^{(k+1)} \cdot \mathbf{d}^{(k)} = 0$.

EXAMPLE 8.2 Analytical Step Size Determination

Let a direction of change for the function

$$f(\mathbf{x}) = 3x_1^2 + 2x_1x_2 + 2x_2^2 + 7 \quad (a)$$

at the point $(1, 2)$ be given as $(-1, -1)$. Compute the step size α_k to minimize $f(\mathbf{x})$ in the given direction.

Solution. For the given point $\mathbf{x}^{(k)} = (1, 2)$, $f(\mathbf{x}^{(k)}) = 22$, and $\mathbf{d}^{(k)} = (-1, -1)$. We first check to see if $\mathbf{d}^{(k)}$ is a direction of descent using Inequality (8.8). The gradient of the function at $(1, 2)$ is given as $\mathbf{c}^{(k)} = (10, 10)$ and $\mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)} = 10(-1) + 10(-1) = -20 < 0$. Therefore, $(-1, -1)$ is a direction of descent. The new point $\mathbf{x}^{(k+1)}$ using Eq. (8.9a) is given as

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{(k+1)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \alpha \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \text{or} \quad x_1^{(k+1)} = 1 - \alpha; \quad x_2^{(k+1)} = 2 - \alpha \quad (b)$$

Substituting these equations into the cost function of Eq. (a), we get

$$f(\mathbf{x}^{(k+1)}) = 3(1 - \alpha)^2 + 2(1 - \alpha)(2 - \alpha) + 2(2 - \alpha)^2 + 7 = 7\alpha^2 - 20\alpha + 22 = f(\alpha) \quad (c)$$

Therefore, along the given direction $(-1, -1)$, $f(\mathbf{x})$ becomes a function of the single variable α . Note from Eq. (c) that $f(0) = 22$, which is the cost function value at the current point, and that $f'(0) = -20 < 0$, which is the slope of $f(\alpha)$ at $\alpha = 0$ (also recall that $f'(0) = \mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)}$). Now using the necessary and sufficient conditions of optimality for $f(\alpha)$, we obtain

$$\frac{df}{d\alpha} = 14\alpha_k - 20 = 0; \quad \alpha_k = \frac{10}{7}; \quad \frac{d^2f}{d\alpha^2} = 14 > 0 \quad (d)$$

Therefore, $\alpha_k = \frac{10}{7}$ minimizes $f(\mathbf{x})$ in the direction $(-1, -1)$. The new point is

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^{(k+1)} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \left(\frac{10}{7}\right) \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} -\frac{3}{7} \\ \frac{4}{7} \end{bmatrix} \quad (e)$$

Substituting the new design $(-\frac{3}{7}, \frac{4}{7})$ into the cost function $f(\mathbf{x})$ we find the new value of the cost function as $\frac{54}{7}$. This is a substantial reduction from the cost function value of 22 at the previous point. Note that Eq. (d) for calculation of step size α can also be obtained by directly using the condition given in Eq. (8.11). Using Eq. (b), the gradient of f at the new design point in terms of α is given as

$$\mathbf{c}^{(k+1)} = (6x_1 + 2x_2, 2x_1 + 4x_2) = (10 - 8\alpha, 10 - 6\alpha) \quad (f)$$

Using the condition of Eq. (8.11), we get $14\alpha - 20 = 0$ which is same as Eq. (d).

8.2.3 Concepts Related to Numerical Methods to Compute Step Size

In Example 8.2, it was possible to simplify expressions and obtain an explicit form for the function $f(\alpha)$. Also, the functional form of $f(\alpha)$ was quite simple. Therefore, it was possible to use the necessary and sufficient conditions of optimality to find the minimum of $f(\alpha)$ and analytically calculate the step size α_k . For many problems, it is not possible to obtain an explicit expression for $f(\alpha)$. Moreover, even if the functional form of $f(\alpha)$ is known, it may be too complicated to lend itself to analytical solution. Therefore, a *numerical method must be used to find α_k* to minimize $f(\mathbf{x})$ in the known direction $\mathbf{d}^{(k)}$.

The numerical line search process is itself iterative, requiring several iterations before a minimum point is reached. Many line search techniques are based on comparing function values at several points along the search direction. Usually, we must make some assumptions on the form of the line search function to compute step size by numerical methods. For example, it must be assumed that a minimum exists and that it is unique in some interval of interest. A function with this property is called the *unimodal function*. Figure 8-4 shows the graph of such a function that decreases continuously until the minimum point is reached. Comparing Figs. 8-3 and 8-4, we observe that $f(\alpha)$ is a unimodal function in some interval. Therefore, it has a unique minimum.

Most one-dimensional search methods assume the line search function to be a unimodal function. This may appear to be a severe restriction on the methods; however, it is not. For functions that are not unimodal, we can think of locating only a local minimum point that is closest to the starting point, i.e., closest to $\alpha = 0$. This is illustrated in Fig. 8-5, where the function $f(\alpha)$ is not unimodal for $0 \leq \alpha \leq \alpha_0$. Points A, B, and C are all local minima. If we restrict α to lie between 0 and $\bar{\alpha}$, however, there is only one local minimum point A because the function $f(\alpha)$ is unimodal for $0 \leq \alpha \leq \bar{\alpha}$. Thus, the assumption of unimodality is not as restrictive as it appears.

The line search problem then is to find α in an interval $0 \leq \alpha \leq \bar{\alpha}$ at which the function $f(\alpha)$ has a global minimum. This statement of the problem, however, requires some modification. Since we are dealing with numerical methods, it is not possible to locate the exact minimum point α^* . In fact, *what we determine is the interval in which the minimum lies*, i.e., some lower and upper limits α_l and α_u for α^* . The interval (α_l, α_u) is called the *interval of uncertainty* and is designated as $I = \alpha_u - \alpha_l$. Most numerical methods iteratively reduce the interval of uncertainty until it satisfies a specified tolerance ε , i.e., $I < \varepsilon$. Once this stopping criterion is satisfied, α^* is taken as $0.5(\alpha_l + \alpha_u)$. Methods based on the preceding philosophy

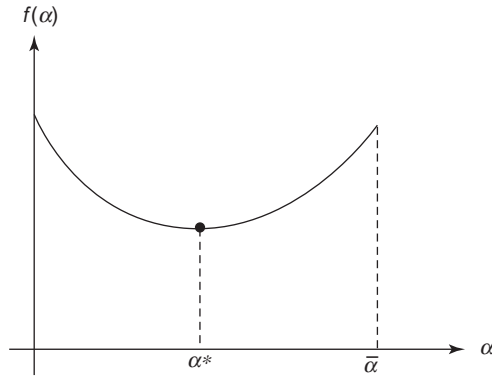


FIGURE 8-4 Unimodal function $f(\alpha)$.

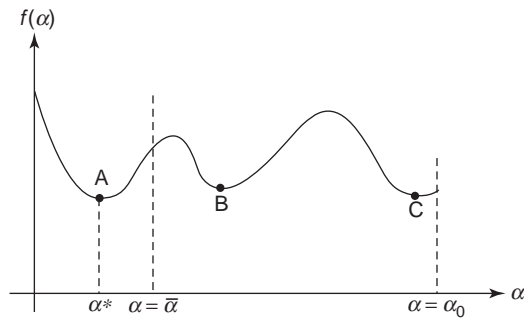


FIGURE 8-5 Nonunimodal function $f(\alpha)$ for $0 \leq \alpha \leq \alpha_0$ (unimodal for $0 \leq \alpha \leq \bar{\alpha}$).

are called *interval reducing methods*. In this chapter, we shall only present methods based on this idea. The *basic procedure for these methods* can be divided into *two phases*. In *phase one*, the location of the minimum point is bracketed and the initial interval of uncertainty is established. In the *second phase*, the interval of uncertainty is refined by eliminating regions that cannot contain the minimum. This is done by computing and comparing function values in the interval of uncertainty. We shall describe the two phases for these methods in more detail in the following subsections.

It is important to note that the performance of most optimization methods depends heavily on the step size calculation procedure. Therefore, it is not surprising that numerous procedures have been developed and evaluated for step size calculation. In the sequel, we describe two rudimentary methods to give the students a flavor of the calculations needed to evaluate a step size. In Chapter 9, some more advanced methods based on the concept of an inaccurate line search are described and discussed.

8.2.4 Equal Interval Search

As mentioned earlier, *the basic idea of any interval reducing method is to reduce successively the interval of uncertainty to a small acceptable value*. To clearly discuss the ideas, we start with a very simple-minded approach called the equal interval search method. The idea is quite elementary as illustrated in Fig. 8-6. In the interval $0 \leq \alpha \leq \bar{\alpha}$, the function $f(\alpha)$ is evaluated at several points using a uniform grid in Phase I. To do this, we select a small number δ and evaluate the function at the α values of $\delta, 2\delta, 3\delta, \dots, q\delta, (q+1)\delta$, and so on

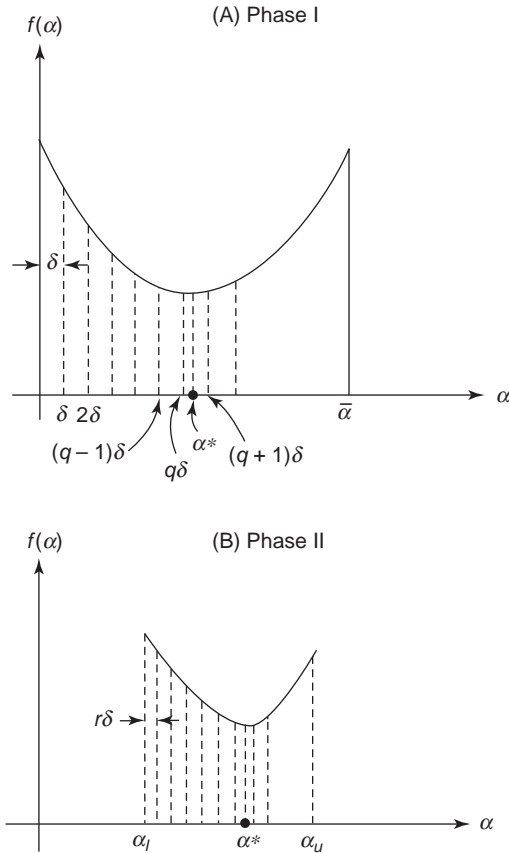


FIGURE 8-6 Equal interval search process. (A) Phase I: Initial bracketing of minimum. (B) Phase II: Reducing the interval of uncertainty.

as shown in Fig. 8-6(A). We compare values of the function at the two successive points, say q and $(q + 1)$. Then, if the function at the point q is larger than that at the next point $(q + 1)$, i.e., $f(q\delta) > f((q + 1)\delta)$ the minimum point has not been surpassed yet. However, if the function has started to increase, i.e.,

$$f(q\delta) < f((q + 1)\delta) \quad (8.12)$$

then the minimum has been surpassed. Note that once Eq. (8.12) is satisfied for points q and $(q + 1)$, the minimum can be between either the points $(q - 1)$ and q or the points q and $(q + 1)$. To account for both possibilities, we take the minimum to lie between the points $(q - 1)$ and $(q + 1)$. Thus, lower and upper limits for the interval of uncertainty are established as

$$\alpha_l = (q - 1)\delta, \quad \alpha_u = (q + 1)\delta, \quad I = \alpha_u - \alpha_l = 2\delta \quad (8.13)$$

Establishment of the lower and upper limits on the minimum value of α indicates end of Phase I. In Phase II, we restart the search process from the lower end of the interval of uncertainty $\alpha = \alpha_l$ with some reduced value for the increment in δ , say $r\delta$, where $r \ll 1$. Then, the preceding process of Phase I is repeated from $\alpha = \alpha_l$ with the reduced δ and the minimum is again bracketed. Now, the interval of uncertainty I is reduced to $2r\delta$. This is illustrated in Fig. 8-6(B). The value of the increment is further reduced, to say $r^2\delta$, and the process is

repeated, until the interval of uncertainty is reduced to an acceptable value ε . Note that the method is *convergent* for unimodal functions and can be easily coded into a computer program.

The efficiency of a method such as the equal interval search depends on the number of function evaluations needed to achieve the desired accuracy. Clearly, this depends on the initial choice for the value of δ . If δ is very small, the process may take many function evaluations to initially bracket the minimum. An advantage of using a smaller δ , however, is that the interval of uncertainty at the end of the Phase I is fairly small. Subsequent improvements for the interval of uncertainty require fewer function evaluations. It is usually advantageous to start with a larger value of δ and quickly bracket the minimum point. Then, the process is continued until the accuracy requirement is satisfied.

8.2.5 Alternate Equal Interval Search

A slightly different computational procedure can be followed to reduce the interval of uncertainty in Phase II once the minimum has been bracketed in Phase I. This procedure is a precursor to the more efficient golden sections search presented in the next section. The procedure is to evaluate the function at two new points, say α_a and α_b in the interval of uncertainty. The points α_a and α_b are located at a distance of $I/3$ and $2I/3$ from the lower limit α_l , respectively, where $I = \alpha_u - \alpha_l$. That is,

$$\alpha_a = \alpha_l + \frac{1}{3}I; \quad \alpha_b = \alpha_l + \frac{2}{3}I = \alpha_u - \frac{1}{3}I$$

This is shown in Fig. 8-7. Next, the function is evaluated at the two new points α_a and α_b . Let these be designated as $f(\alpha_a)$ and $f(\alpha_b)$. Now, the following two conditions must be checked:

1. If $f(\alpha_a) < f(\alpha_b)$, then the minimum lies between α_l and α_b . The right one-third interval between α_b and α_u is discarded. New limits for the interval of uncertainty are $\alpha'_l = \alpha_l$ and $\alpha'_u = \alpha_b$ (the prime on α is used to indicate revised limits for the interval of uncertainty). Therefore, the reduced interval of uncertainty is $I' = \alpha'_u - \alpha'_l = \alpha_b - \alpha_l$. The procedure is repeated with the new limits.
2. If $f(\alpha_a) > f(\alpha_b)$, then the minimum lies between α_a and α_u . The interval between α_l and α_a is discarded. The procedure is repeated with $\alpha'_l = \alpha_a$ and $\alpha'_u = \alpha_u$ ($I' = \alpha'_u - \alpha'_l$).

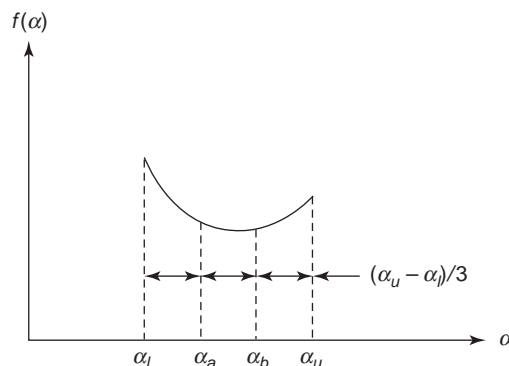


FIGURE 8-7 An alternate equal interval search process.

With the preceding calculations, the interval of uncertainty is reduced to $I' = 2I/3$ after every set of two function evaluations. The entire process is continued until the interval of uncertainty is reduced to an acceptable value.

8.2.6 Golden Section Search

Golden section search is an improvement over the alternate equal interval search and is one of the better methods in the class of interval reducing methods. The basic idea of the method is still the same: evaluate the function at predetermined points, compare them to bracket the minimum in Phase I, and then converge on the minimum point in Phase II. The method uses fewer function evaluations to reach the minimum point compared with other similar methods. The number of function evaluations is reduced during both the phases, the initial bracketing phase as well as the interval reducing phase.

Initial Bracketing of Minimum—Phase I In the equal interval methods, the initially selected increment δ is kept fixed to bracket the minimum initially. This can be an inefficient process if δ happens to be a small number. An alternate procedure is to vary the increment at each step, i.e., multiply it by a constant $r > 1$. This way initial bracketing of the minimum is rapid; however, the length of the initial interval of uncertainty is increased. The *golden section* search procedure is such a *variable interval search method*. In the method the value of r is not selected arbitrarily. It is selected as the *golden ratio*, which can be derived as 1.618 in several different ways. One derivation is based on the *Fibonacci sequence* defined as

$$F_0 = 1; \quad F_1 = 1; \quad F_n = F_{n-1} + F_{n-2}, \quad n = 2, 3, \dots \quad (a)$$

Any number of the Fibonacci sequence for $n > 1$ is obtained by adding the previous two numbers, so the sequence is given as 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, The sequence has the property,

$$\frac{F_n}{F_{n-1}} \rightarrow 1.618 \quad \text{as} \quad n \rightarrow \infty \quad (b)$$

That is, as n becomes large, the ratio between two successive numbers F_n and F_{n-1} in the Fibonacci sequence reaches a constant value of 1.618 or $(\sqrt{5} + 1)/2$. This golden ratio has many other interesting properties that will be exploited in the one-dimensional search procedure. One property is that $1/1.618 = 0.618$.

Figure 8-8 illustrates the process of initially bracketing the minimum using a sequence of larger increments based on the golden ratio. In the figure, starting at $q = 0$, we evaluate $f(\alpha)$ at $\alpha = \delta$, where $\delta > 0$ is a small number. We check to see if the value $f(\delta)$ is smaller than the value $f(0)$. If it is, we then take an increment of 1.618δ in the step size (i.e., the increment is 1.618 times the previous increment δ). This way we evaluate the function at the following points and compare them:

$$\begin{aligned} q = 0; & \quad \alpha_0 = \delta \\ q = 1; & \quad \alpha_1 = \delta + 1.618\delta = 2.618\delta = \sum_{j=0}^1 \delta(1.618)^j \\ q = 2; & \quad \alpha_2 = 2.618\delta + 1.618(1.618\delta) = 5.236\delta = \sum_{j=0}^2 \delta(1.618)^j \\ q = 3; & \quad \alpha_3 = 5.236\delta + 1.618^3\delta = 9.472\delta = \sum_{j=0}^3 \delta(1.618)^j \\ & \quad \cdot \\ & \quad \cdot \\ & \quad \cdot \end{aligned}$$

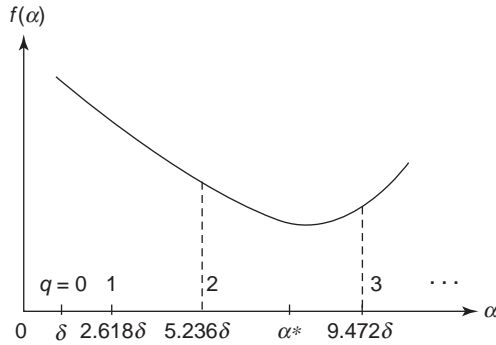


FIGURE 8-8 Initial bracketing of the minimum point in the golden section method.

In general, we continue to evaluate the function at the points

$$\alpha_q = \sum_{j=0}^q \delta(1.618)^j; \quad q = 0, 1, 2, \dots \quad (8.14)$$

Let us assume that the function at α_{q-1} is smaller than that at the previous point α_{q-2} and the next point α_q , i.e.,

$$f(\alpha_{q-1}) < f(\alpha_{q-2}) \quad \text{and} \quad f(\alpha_{q-1}) < f(\alpha_q) \quad (8.15)$$

Therefore, the minimum point has been surpassed. Actually the minimum point lies between the previous two intervals, i.e., between α_q and α_{q-2} , as in the equal interval search. Therefore, *upper and lower limits on the interval of uncertainty* are

$$\alpha_u = \alpha_q = \sum_{j=0}^q \delta(1.618)^j; \quad \alpha_l = \alpha_{q-2} = \sum_{j=0}^{q-2} \delta(1.618)^j \quad (8.16)$$

Thus, the *initial interval of uncertainty* is calculated as

$$\begin{aligned} I = \alpha_u - \alpha_l &= \sum_{j=0}^q \delta(1.618)^j - \sum_{j=0}^{q-2} \delta(1.618)^j = \delta(1.618)^{q-1} + \delta(1.618)^q \\ &= \delta(1.618)^{q-1} (1 + 1.618) = 2.618(1.618)^{q-1} \delta \end{aligned} \quad (8.17)$$

Reduction of Interval of Uncertainty—Phase II The next task is to start reducing the interval of uncertainty by evaluating and comparing functions at some points in the established interval of uncertainty I . The method uses two function values within the interval I , just as in the alternate equal interval search of Fig. 8-7. However, the points α_a and α_b are not located at $I/3$ from either end of the interval of uncertainty. Instead, they are located at a distance of $0.382I$ (or $0.618I$) from either end. The factor 0.382 is related to the golden ratio as we shall see in the following.

To see how the factor 0.618 is determined, consider two points symmetrically located from either end as shown in Fig. 8-9(A)—points α_a and α_b are located at a distance of τI from either end of the interval. Comparing functions values at α_a and α_b , either the left (α_l, α_a) or the right (α_b, α_u) portion of the interval gets discarded because the minimum cannot lie there. Let us assume that the right portion gets discarded as shown in Fig. 8-9(B), so α'_l and α'_u are

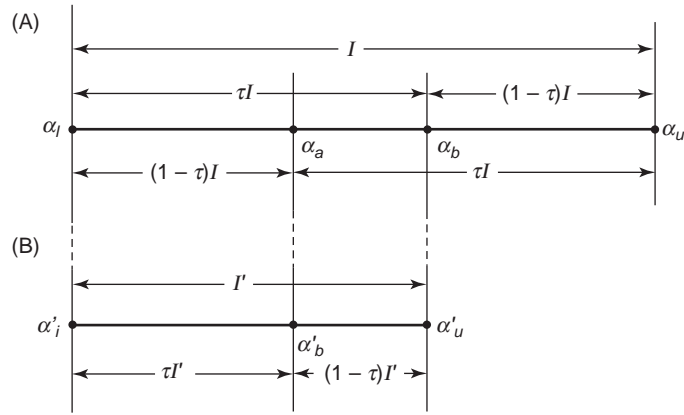


FIGURE 8-9 Golden section partition.

the new lower and upper bounds on the minimum. The new interval of uncertainty is $I' = \tau I$. There is one point in the new interval at which the function value is known. It is required that this point be located at a distance of $\tau I'$ from the left end; therefore, $\tau I' = (1 - \tau)I$. Since $I' = \tau I$, this gives the equation $\tau^2 + \tau - 1 = 0$. The positive root of this equation is $\tau = (-1 + \sqrt{5})/2 = 0.618$. Thus the two points are located at a distance of $0.618I$ or $0.382I$ from either end of the interval.

The golden section search can be initiated once the initial interval of uncertainty is known. If the initial bracketing is done using the variable step increment (with a factor of 1.618, which is $1/0.618$), then the function value at one of the points α_{q-1} is already known. It turns out that α_{q-1} is automatically the point α_a . This can be seen by multiplying the initial interval I in Eq. (8.17) by 0.382. If the preceding procedure is not used to initially bracket the minimum, then the points α_a and α_b will have to be calculated by the golden section procedure.

Algorithm for One-Dimensional Search by Golden Sections Find α to minimize $f(\alpha)$.

- Step 1.* For a chosen small number δ , let q be the smallest integer to satisfy Eq. (8.15) where α_q , α_{q-1} , and α_{q-2} are calculated from Eq. (8.14). The upper and lower bounds on α^* (the optimum value for α) are given by Eq. (8.16).
- Step 2.* Compute $f(\alpha_b)$, where $\alpha_b = \alpha_l + 0.618I$ (the interval of uncertainty $I = \alpha_u - \alpha_l$). Note that, at the first iteration, $\alpha_a = \alpha_l + 0.382I = \alpha_{q-1}$, and so $f(\alpha_a)$ is already known.
- Step 3.* Compare $f(\alpha_a)$ and $f(\alpha_b)$, and go to (i), (ii), or (iii).
- (i) If $f(\alpha_a) < f(\alpha_b)$, then minimum point α^* lies between α_l and α_b , i.e., $\alpha_l \leq \alpha^* \leq \alpha_b$. The new limits for the reduced interval of uncertainty are $\alpha'_l = \alpha_l$ and $\alpha'_u = \alpha_b$. Also, $\alpha'_b = \alpha_a$. Compute $f(\alpha'_a)$, where $\alpha'_a = \alpha'_l + 0.382(\alpha'_u - \alpha'_l)$ and go to Step 4.
 - (ii) If $f(\alpha_a) > f(\alpha_b)$, then minimum point α^* lies between α_a and α_u , i.e., $\alpha_a \leq \alpha^* \leq \alpha_u$. Similar to the procedure in Step 3(i), let $\alpha'_l = \alpha_a$ and $\alpha'_u = \alpha_u$, so that $\alpha'_a = \alpha_b$. Compute $f(\alpha'_b)$, where $\alpha'_b = \alpha'_l + 0.618(\alpha'_u - \alpha'_l)$ and go to Step 4.
 - (iii) If $f(\alpha_a) = f(\alpha_b)$, let $\alpha_l = \alpha_a$ and $\alpha_u = \alpha_b$ and return to Step 2.
- Step 4.* If the new interval of uncertainty $I' = \alpha'_u - \alpha'_l$ is small enough to satisfy a stopping criterion (i.e., $I' < \epsilon$), let $\alpha^* = (\alpha'_u + \alpha'_l)/2$ and stop. Otherwise, delete the primes on α'_l , α'_a , and α'_b and return to Step 3.

Example 8.3 illustrates the golden sections method for step size calculation.

EXAMPLE 8.3 Minimization of a Function by Golden Section Search

Consider the function $f(\alpha) = 2 - 4\alpha + e^\alpha$. Use golden section search to find the minimum within an accuracy of $\varepsilon = 0.001$. Use $\delta = 0.5$.

Solution. Analytically, the solution is $\alpha^* = 1.3863$, $f(\alpha^*) = 0.4548$. In the golden section search, we need to first bracket the minimum point (Phase I) and then iteratively reduce the interval of uncertainty (Phase II). Table 8-1 shows various iterations of the method. In Phase I, the minimum point is bracketed in only four iterations as shown in the first part of the table. The initial interval of uncertainty is calculated as $I = (\alpha_u - \alpha_l) = 2.618034 - 0.5 = 2.118034$ since $f(2.618034) > f(1.309017)$ in Table 8-1. Note that this interval would be larger than the one obtained using equal interval searching.

Now, to reduce the interval of uncertainty in Phase II, let us calculate α_b as $(\alpha_l + 0.618I)$ or $\alpha_b = \alpha_l - 0.382I$ (calculations are shown in the second part of Table 8-1). Note that α_u and $f(\alpha_u)$ are already known and need no further calculation. This is the main advantage of the golden section search; only one additional function evaluation is needed in the interval of uncertainty in each iteration, compared with the two function evaluations needed for the alternate equal interval search. We calculate $\alpha_b = 1.809017$ and $f(\alpha_b) = 0.868376$. Note that the new calculation of the function is shown in boldface for each iteration. Since $f(\alpha_u) < f(\alpha_b)$, new limits for the reduced interval

TABLE 8-1 Golden Section Search for $f(\alpha) = 2 - 4\alpha + e^\alpha$ of Example 8.3

Phase 1: Initial bracketing of minimum

No., q	Trial step, α	Function value, $f(\alpha)$
1	0.000000	3.000000
2	$\alpha_l \rightarrow 0.500000$	1.648721
3	1.309017	0.466464
4	$\alpha_u \rightarrow 2.618034$	5.236610

Phase 2: Reducing interval of uncertainty

No.	$\alpha_l [f(\alpha_l)]$	$\alpha_a [f(\alpha_a)]$	$\alpha_b [f(\alpha_b)]$	$\alpha_u [f(\alpha_u)]$	I
1	0.500000 [1.648721] ↓	1.309017 [0.466464] ↓	1.809017 [0.868376] ↓	2.618034 [5.236610]	2.118034
2	0.500000 [1.648721]	1.000000 ✓ [0.718282]	1.309017 ✓ [0.466464]	1.809017 [0.868376] ↓	1.309017
3	1.000000 [0.718282]	1.309017 [0.466464]	1.500000 [0.481689]	1.809017 [0.868376]	0.809017
—	—	—	—	—	—
—	—	—	—	—	—
16	1.385438 [0.454824]	1.386031 [0.454823]	1.386398 [0.454823]	1.386991 [0.454824]	0.001553
17	1.386031 [0.454823]	1.386398 [0.454823]	1.386624 [0.454823]	1.386991 [0.454823]	0.000960

$\alpha = 0.5(1.386398 + 1.386624) = 1.386511$; $f(\alpha^*) = 0.454823$.

Note: The new calculation for each iteration is shown as boldfaced and shaded; the arrows indicate direction of transfer of data.

of uncertainty are $\alpha'_i = 0.5$ and $\alpha'_{ii} = 1.809017$. Also, $\alpha'_b = 1.309017$ at which the function value is already known. We need to compute only $f(\alpha'_a)$ where $\alpha'_a = \alpha'_i + 0.382(\alpha'_{ii} - \alpha'_i) = 1.000$. Further refinement of the interval of uncertainty is repetitive and can be accomplished by writing a computer program.

A subroutine GOLD implementing the golden section search procedure is given in Appendix D. The minimum for the function f is obtained at $\alpha^* = 1.386511$ with $f(\alpha^*) = 0.454823$ in 22 function evaluations as shown in Table 8-1. The number of function evaluations is a measure of efficiency of an algorithm. The problem was also solved using the equal interval search and 37 function evaluations were needed to obtain the same solution. This verifies our earlier observation that golden section search is a better method for a specified accuracy and initial step length.

It may appear that if the initial step length δ is too large in the equal interval or golden section method, the line search fails, i.e., $f(\delta) > f(0)$. Actually, it indicates that initial δ is not proper and needs to be reduced until $f(\delta) < f(0)$. With this procedure, convergence of the method can be numerically enforced. This numerical procedure has been implemented in the GOLD subroutine given in Appendix D.

8.3 Search Direction Determination: Steepest Descent Method

Thus far we have assumed that a search direction in the design space was known and we have tackled the problem of step size determination. In this section and the next, we shall address the question of how to determine the search direction \mathbf{d} . The *basic requirement for \mathbf{d}* is that the cost function be reduced if we make a small move along \mathbf{d} ; that is, the descent condition of Eq. (8.8) be satisfied. This will be called the *descent direction*.

Several methods are available for determining a descent direction for unconstrained optimization problems. The *steepest descent method* or the *gradient method* is the simplest, the oldest, and probably the best known numerical method for unconstrained optimization. The philosophy of the method, introduced by Cauchy in 1847, is to find the direction \mathbf{d} at the current iteration in which the cost function $f(\mathbf{x})$ decreases most rapidly, at least locally. Because of this philosophy, the method is called the *steepest descent* search technique. Also, properties of the gradient of the cost function are used in the iterative process, which is the reason for its alternate name: the *gradient method*. The steepest descent method is a *first-order method* since only the gradient of the cost function is calculated and used to evaluate the search direction. In the next chapter, we shall discuss *second-order methods* in which the Hessian of the function will be used in determining the search direction.

The gradient of a scalar function $f(x_1, x_2, \dots, x_n)$ was defined in Chapter 4 as the column vector:

$$\mathbf{c} = \nabla f = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \dots \quad \frac{\partial f}{\partial x_n} \right]^T \quad (8.18)$$

To simplify the notation, we shall use vector \mathbf{c} to represent gradient of the cost function $f(\mathbf{x})$; that is, $c_i = \partial f / \partial x_i$. We shall use a superscript to denote the point at which this vector is calculated, as

$$\mathbf{c}^{(k)} = \mathbf{c}(\mathbf{x}^{(k)}) = \left[\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_i} \right]^T \quad (8.19)$$

The *gradient vector has several properties* that are used in the steepest descent method. These will be discussed in the next chapter in more detail. The most important property is that the *gradient at a point \mathbf{x} points in the direction of maximum increase in the cost func-*

tion. Thus the direction of maximum decrease is opposite to that, i.e., negative of the gradient vector. Any small move in the negative gradient direction will result in the maximum local rate of decrease in the cost function. The negative gradient vector then represents a *direction of steepest descent* for the cost function and is written as

$$\mathbf{d} = -\mathbf{c}, \quad \text{or} \quad d_i = -c_i = -\frac{\partial f}{\partial x_i}; \quad i = 1 \text{ to } n \quad (8.20)$$

Equation (8.20) gives a direction of change in the design space for use in Eq. (8.4). Based on the preceding discussion, the *steepest descent algorithm* is stated as follows:

- Step 1.* Estimate a starting design $\mathbf{x}^{(0)}$ and set the iteration counter $k = 0$. Select a convergence parameter $\varepsilon > 0$.
- Step 2.* Calculate the gradient of $f(\mathbf{x})$ at the point $\mathbf{x}^{(k)}$ as $\mathbf{c}^{(k)} = \nabla f(\mathbf{x}^{(k)})$.
- Step 3.* Calculate $\|\mathbf{c}^{(k)}\|$. If $\|\mathbf{c}^{(k)}\| < \varepsilon$, then stop the iterative process because $\mathbf{x}^* = \mathbf{x}^{(k)}$ is a minimum point. Otherwise, continue.
- Step 4.* Let the search direction at the current point $\mathbf{x}^{(k)}$ be $\mathbf{d}^{(k)} = -\mathbf{c}^{(k)}$.
- Step 5.* Calculate a step size α_k that minimizes $f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$. Any one-dimensional search algorithm may be used to determine α_k .
- Step 6.* Update the design as $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)}$. Set $k = k + 1$, and go to Step 2.

The basic idea of the steepest descent method is quite simple. We start with an initial estimate for the minimum design. The direction of steepest descent is computed at that point. If the direction is nonzero, we move as far as possible along it to reduce the cost function. At the new design point, we calculate the steepest descent direction again and repeat the entire process. Note that since $\mathbf{d} = -\mathbf{c}$, the *descent condition of inequality* (8.8) is always satisfied as $\mathbf{c} \cdot \mathbf{d} = -\|\mathbf{c}\|^2 < 0$. Examples 8.4 and 8.5 illustrate the calculations involved in the steepest descent method.

EXAMPLE 8.4 Use of Steepest Descent Algorithm

$$\text{Minimize } f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1x_2 \quad (a)$$

using the steepest descent method starting from the point (1, 0).

Solution. To solve the problem, we follow the steps of the steepest descent algorithm.

1. The starting design is given as $\mathbf{x}^{(0)} = (1, 0)$.
2. $\mathbf{c}^{(0)} = (2x_1 - 2x_2, 2x_2 - 2x_1) = (2, -2)$.
3. $\|\mathbf{c}^{(0)}\| = 2\sqrt{2} \neq 0$
4. Set $\mathbf{d}^{(0)} = -\mathbf{c}^{(0)} = (-2, 2)$.
5. Calculate α to minimize $f(\mathbf{x}^{(0)} + \alpha\mathbf{d}^{(0)})$ where $\mathbf{x}^{(0)} + \alpha\mathbf{d}^{(0)} = (1 - 2\alpha, 2\alpha)$:

$$\begin{aligned} f(\mathbf{x}^{(0)} + \alpha\mathbf{d}^{(0)}) &= (1 - 2\alpha)^2 + (2\alpha)^2 + (2\alpha)^2 - 2(1 - 2\alpha)(2\alpha) \\ &= 16\alpha^2 - 8\alpha + 1 = f(\alpha) \end{aligned} \quad (b)$$

Since this is a simple function of α , we can use necessary and sufficient conditions to solve for the optimum step length. In general, a numerical one-dimensional search will have to be used to calculate α . Using the analytic approach to solve for optimum α , we get

$$\frac{df(\alpha)}{d\alpha} = 0; \quad 32\alpha - 8 = 0 \quad \text{or} \quad \alpha_0 = 0.25 \quad (\text{c})$$

$$\frac{d^2 f(\alpha)}{d\alpha^2} = 32 > 0. \quad (\text{d})$$

Therefore, the sufficiency condition for a minimum for $f(\alpha)$ is satisfied.

6. Updating the design $(\mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)})$: $x_1^{(1)} = 1 - 0.25(2) = 0.5$, $x_2^{(1)} = 0 + 0.25(2) = 0.5$. Solving for $\mathbf{c}^{(1)}$ from the expression in Step 2, we see that $\mathbf{c}^{(1)} = (0, 0)$, which satisfies the stopping criterion. Therefore, $(0.5, 0.5)$ is a minimum point for $f(\mathbf{x})$ and $f^* = 0$.

The preceding problem is quite simple and an optimum point is obtained in only one iteration. This is because the condition number of the Hessian of the cost function is 1 (condition number is a scalar associated with the given matrix; refer to Section B.7 in Appendix B). In such a case, the steepest descent method converges in just one iteration with any starting point. In general, the algorithm will require several iterations before an acceptable optimum is reached.

EXAMPLE 8.5 Use of Steepest Descent Algorithm

$$\text{Minimize } f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3 \quad (\text{a})$$

using the steepest descent method with a starting design as $(2, 4, 10)$. Select the convergence parameter ε as 0.005. Perform a line search by golden section search with initial step length $\delta = 0.05$ and an accuracy of 0.0001.

Solution.

1. The starting point is set as $\mathbf{x}^{(0)} = (2, 4, 10)$.
2. $\mathbf{c} = \nabla f = (2x_1 + 2x_2, 4x_2 + 2x_1 + 2x_3, 4x_3 + 2x_2)$; $\mathbf{c}^{(0)} = (12, 40, 48)$.
3. $\|\mathbf{c}^{(0)}\| = \sqrt{4048} = 63.6 > \varepsilon$ (continue).
4. $\mathbf{d}^{(0)} = -\mathbf{c}^{(0)} = (-12, -40, -48)$.
5. Calculate α_0 by golden section search to minimize $f(\mathbf{x}^{(0)} + \alpha \mathbf{d}^{(0)})$; $\alpha_0 = 0.1587$.
6. Update the design as $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)} = (0.0956, -2.348, 2.381)$. At the new design, $\mathbf{c}^{(1)} = (-4.5, -4.438, 4.828)$, $\|\mathbf{c}^{(1)}\| = 7.952 > \varepsilon$.

Note that $\mathbf{c}^{(1)} \cdot \mathbf{d}^{(0)} = 0$, which verifies the exact line search termination criterion given in Eq. (8.11). The steps in steepest descent algorithm should be repeated until the convergence criterion is satisfied. Appendix D contains the computer program and user supplied subroutines FUNCT and GRAD to implement steps of the steepest descent algorithm. The optimum results for the problem with the program are given in Table 8-2. The true optimum cost function value is 0.0 and the optimum point is $(0, 0, 0)$.

Note that large numbers of iterations and function evaluations are needed to reach the optimum.

TABLE 8-2 Optimum Solution for Example 8.5 with Steepest Descent Method:

$$f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$$

Starting values of design variables: 2, 4, 10

Optimum design variables: 8.04787E-03, -6.81319E-03, 3.42174E-03

Optimum cost function value: 2.473 47E-05

Norm of gradient of the cost function at optimum: 4.970 71E-03

Number of iterations: 40

Total number of function evaluations: 753

Although the method of steepest descent is quite simple and robust (it is convergent), it has some drawbacks. These are:

1. Even if convergence of the method is guaranteed, a large number of iterations may be required for the minimization of even positive definite quadratic forms, i.e., the method can be quite slow to converge to the minimum point.
2. Information calculated at the previous iterations is not used. Each iteration is started independent of others, which is inefficient.
3. Only first-order information about the function is used at each iteration to determine the search direction. This is one reason that convergence of the method is slow. It can further deteriorate if an inaccurate line search is used. Moreover, the rate of convergence depends on the condition number of the Hessian of the cost function at the optimum point. If the condition number is large, the rate of convergence of the method is slow.
4. Practical experience with the method has shown that a substantial decrease in the cost function is achieved in the initial few iterations and then it decreases quite slowly in later iterations.
5. The direction of steepest descent (direction of most rapid decrease in the cost function) may be good in a local sense (in a small neighborhood) but not in a global sense.

8.4 Search Direction Determination: Conjugate Gradient Method

There are many optimization methods based on the concept of conjugate gradients; however, we shall only present a method due to Fletcher and Reeves (1964). The conjugate gradient method is a very simple and effective modification of the steepest descent method. It will be shown in the next chapter that the steepest descent directions at two consecutive steps are orthogonal to each other. This tends to slow down the steepest descent method although it is convergent. The conjugate gradient directions are not orthogonal to each other. Rather, these directions tend to cut diagonally through the orthogonal steepest descent directions. Therefore, they improve the rate of convergence of the steepest descent method considerably.

Actually, the *conjugate gradient directions* $\mathbf{d}^{(i)}$ are orthogonal with respect to a symmetric and positive definite matrix \mathbf{A} , i.e., $\mathbf{d}^{(i)T} \mathbf{A} \mathbf{d}^{(j)} = 0$ for all i and j , $i \neq j$. The *conjugate gradient algorithm* is stated as follows:

Step 1. Estimate a starting design as $\mathbf{x}^{(0)}$. Set the iteration counter $k = 0$. Select the convergence parameter ε . Calculate

$$\mathbf{d}^{(0)} = -\mathbf{c}^{(0)} = -\nabla f(\mathbf{x}^{(0)}) \quad (8.21a)$$

Check stopping criterion. If $\|\mathbf{c}^{(0)}\| < \varepsilon$, then stop. Otherwise, go to Step 4 (note that Step 1 of the conjugate gradient and the steepest descent methods is the same).

Step 2. Compute the gradient of the cost function as $\mathbf{c}^{(k)} = \nabla f(\mathbf{x}^{(k)})$.

Step 3. Calculate $\|\mathbf{c}^{(k)}\|$. If $\|\mathbf{c}^{(k)}\| < \varepsilon$, then stop; otherwise continue.

Step 4. Calculate the new conjugate direction as

$$\mathbf{d}^{(k)} = -\mathbf{c}^{(k)} + \beta_k \mathbf{d}^{(k-1)}; \quad \beta_k = (\|\mathbf{c}^{(k)}\| / \|\mathbf{c}^{(k-1)}\|)^2 \quad (8.21b)$$

Step 5. Compute a step size $\alpha_k = \alpha$ to minimize $f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$.

Step 6. Change the design as follows, set $k = k + 1$ and go to Step 2.

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)} \quad (8.22)$$

Note that the conjugate direction in Eq. (8.21b) satisfies the descent condition of Inequality (8.8). This can be shown by substituting $\mathbf{d}^{(k)}$ from Eq. (8.21b) into Inequality (8.8) and using the step size determination condition given in Eq. (8.11). The first step of the conjugate gradient method is just the steepest descent step. The only difference between the conjugate gradient and steepest descent methods is in Eq. (8.21b). In this step the current steepest descent direction is modified by adding a scaled direction used in the previous iteration. The scale factor is determined using lengths of the gradient vector at the two iterations as shown in Eq. (8.21b). Thus, the conjugate direction is nothing but a deflected steepest descent direction. This is an extremely simple modification that requires little additional calculation. It is, however, very effective in substantially improving the rate of convergence of the steepest descent method. Therefore, *the conjugate gradient method should always be preferred over the steepest descent method*. In the next chapter an example is discussed that compares the rate of convergence of the steepest descent, conjugate gradient, and Newton's methods. We shall see there that the method performs quite well compared with the other two methods.

The conjugate gradient algorithm finds the minimum in n iterations for positive definite quadratic functions having n design variables. For general functions, if the minimum has not been found by then, it is recommended that the iterative process should be restarted every $(n + 1)$ iterations for computational stability. That is, set $\mathbf{x}^{(0)} = \mathbf{x}^{(n+1)}$ and restart the process from Step 1 of the algorithm. The algorithm is very simple to program and works very well for general unconstrained minimization problems. Example 8.6 illustrates the calculations involved in the conjugate gradient method.

EXAMPLE 8.6 Use of Conjugate Gradient Algorithm

Consider the problem solved in Example 8.5: minimize

$$f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3 \quad (\text{a})$$

Carry out two iterations of the conjugate gradient method starting from the design (2, 4, 10).

Solution. The first iteration of the conjugate gradient method is the same as given in Example 8.5:

$$\mathbf{c}^{(0)} = (12, 40, 48); \quad \|\mathbf{c}^{(0)}\| = 63.6, \quad f(\mathbf{x}^{(0)}) = 332.0 \quad (\text{b})$$

$$\mathbf{x}^{(1)} = (0.0956, -2.348, 2.381) \quad (\text{c})$$

The second iteration starts from Step 2 of the conjugate gradient algorithm:

$$2. \quad \mathbf{c}^{(1)} = (-4.5, -4.438, 4.828), \quad f(\mathbf{x}^{(1)}) = 10.75 \quad (\text{d})$$

$$3. \quad \|\mathbf{c}^{(1)}\| = 7.952 > \varepsilon, \text{ so continue.}$$

$$4. \quad \beta_1 = \left[\frac{\|\mathbf{c}^{(1)}\|}{\|\mathbf{c}^{(0)}\|} \right]^2 = (7.952/63.3)^2 = 0.015633 \quad (\text{e})$$

$$\mathbf{d}^{(1)} = -\mathbf{c}^{(1)} + \beta_1 \mathbf{d}^{(0)} = \begin{bmatrix} 4.500 \\ 4.438 \\ -4.828 \end{bmatrix} + (0.015633) \begin{bmatrix} -12 \\ -40 \\ -48 \end{bmatrix} = \begin{bmatrix} 4.31241 \\ 3.81268 \\ -5.57838 \end{bmatrix} \quad (\text{f})$$

5. Step size in the direction $\mathbf{d}^{(1)}$ is calculated as $\alpha = 0.3156$.

$$6. \quad \text{The design is updated as } \mathbf{x}^{(2)} = \begin{bmatrix} 0.0956 \\ -2.348 \\ 2.381 \end{bmatrix} + \alpha \begin{bmatrix} 4.31241 \\ 3.81268 \\ -5.57838 \end{bmatrix} = \begin{bmatrix} 1.4566 \\ -1.1447 \\ 0.6205 \end{bmatrix} \quad (\text{g})$$

Calculating the gradient at this point, we get $\mathbf{c}^{(2)} = (0.6238, -0.4246, 0.1926)$. $\|\mathbf{c}^{(2)}\| = 0.7788 > \varepsilon$, so we need to continue the iterations. Note that $\mathbf{c}^{(2)} \cdot \mathbf{d}^{(1)} = 0$.

The problem is solved using the conjugate gradient method available in the IDESIGN software with $\varepsilon = 0.005$ (Arora and Tseng, 1987a,b). Table 8-3 summarizes performance results for the method. It can be seen that a very precise optimum is obtained in only 4 iterations and 10 function evaluations. Comparing these with the steepest descent method results given in Table 8-2, we conclude that the conjugate gradient method is superior for this example.

TABLE 8-3 Optimum Solution for Example 8.6 with the Conjugate Gradient Method:
 $f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$

Starting values of design variables:	2, 4, 10
Optimum design variables:	-6.4550E-10, -5.8410E-10, 1.3150E-10.
Optimum cost function value:	6.8520E-20.
Norm of the gradient at optimum:	3.0512E-05.
Number of iterations:	4
Number of function evaluations:	10

Example 8.7 illustrates the use of Excel Solver to solve unconstrained optimization problems.

EXAMPLE 8.7 Use of Excel Solver

Solve the problem of Example 8.6 using Solver in Excel.

Solution. Figure 8-10 shows the worksheet and the Solver dialog box for the problem. The worksheet for the problem can be prepared in several different ways as explained earlier in Chapters 4 and 6. For the present example, cell D9 defines the final expression for the cost function. Once the worksheet has been prepared, Solver is invoked under the Tools tab, and the “Options” button is used to invoke the conjugate gradient method. The forward finite difference option is selected for calculation of the gradient of the cost function. The algorithm converges to the solution reported in Table 8-3 in five iterations.

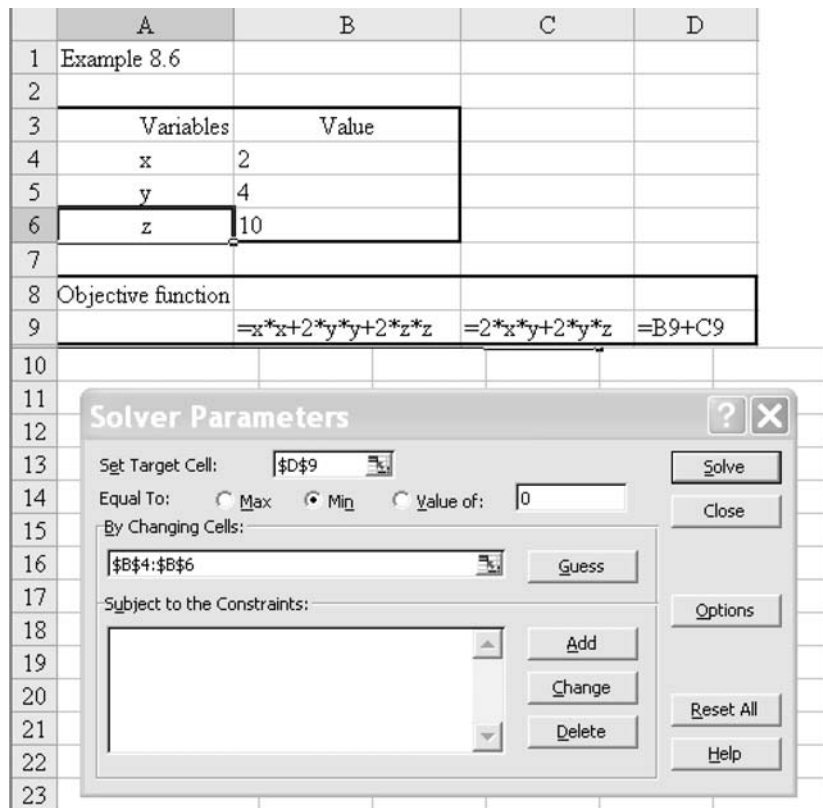


FIGURE 8-10 Excel worksheet and Solver dialog box for Example 8.7.

Exercises for Chapter 8

Section 8.1 General Concepts Related to Numerical Algorithms

8.1 Answer True or False.

1. All optimum design algorithms require a starting point to initiate the iterative process.
2. A vector of design changes must be computed at each iteration of the iterative process.
3. The design change calculation can be divided into step size determination and direction finding subproblems.
4. The search direction requires evaluation of the gradient of the cost function.
5. Step size along the search direction is always negative.
6. Step size along the search direction can be zero.
7. In unconstrained optimization, the cost function can increase for an arbitrary small step along the descent direction.
8. A descent direction always exists if the current point is not a local minimum.
9. In unconstrained optimization, a direction of descent can be found at a point where the gradient of the cost function is zero.
10. The descent direction makes an angle of $0-90^\circ$ with the gradient of the cost function.

Determine if the given direction at the point is that of descent for the following functions (show all the calculations).

8.2 $f(\mathbf{x}) = 3x_1^2 + 2x_1 + 2x_2^2 + 7$; $\mathbf{d} = (-1, 1)$ at $\mathbf{x} = (2, 1)$

8.3 $f(\mathbf{x}) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 4$; $\mathbf{d} = (2, 1)$ at $\mathbf{x} = (1, 1)$

8.4 $f(\mathbf{x}) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$; $\mathbf{d} = (-3, 10, -12)$ at $\mathbf{x} = (1, 2, 3)$

8.5 $f(\mathbf{x}) = 0.1x_1^2 + x_2^2 - 10$; $\mathbf{d} = (1, 2)$ at $\mathbf{x} = (4, 1)$

8.6 $f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2$; $\mathbf{d} = (2, 3)$ at $\mathbf{x} = (4, 3)$

8.7 $f(\mathbf{x}) = 10(x_2 - x_1^2)^2 + (1 - x_1)^2$; $\mathbf{d} = (162, -40)$ at $\mathbf{x} = (2, 2)$

8.8 $f(\mathbf{x}) = (x_1 - 2)^2 + x_2^2$; $\mathbf{d} = (-2, 2)$ at $\mathbf{x} = (1, 1)$

8.9 $f(\mathbf{x}) = 0.5x_1^2 + x_2^2 - x_1x_2 - 7x_1 - 7x_2$; $\mathbf{d} = (7, 6)$ at $\mathbf{x} = (1, 1)$

8.10 $f(\mathbf{x}) = (x_1 + x_2)^2 + (x_2 + x_3)^2$; $\mathbf{d} = (4, 8, 4)$ at $\mathbf{x} = (1, 1, 1)$

8.11 $f(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2$; $\mathbf{d} = (2, 4, -2)$ at $\mathbf{x} = (1, 2, -1)$

8.12 $f(\mathbf{x}) = (x_1 + 3x_2 + x_3)^2 + 4(x_1 - x_2)^2$; $\mathbf{d} = (-2, -6, -2)$ at $\mathbf{x} = (-1, -1, -1)$

8.13 $f(\mathbf{x}) = 9 - 8x_1 - 6x_2 - 4x_3 - 2x_1^2 + 2x_2^2 + x_3^2 + 2x_1x_2 + 2x_2x_3$; $\mathbf{d} = (-2, 2, 0)$ at $\mathbf{x} = (1, 1, 1)$

8.14 $f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 + (x_4 - 4)^2$; $\mathbf{d} = (2, -2, 2, -2)$ at $\mathbf{x} = (2, 1, 4, 3)$

Section 8.2 Basic Ideas and Algorithms for Step Size Determination

8.15 Answer True or False.

1. Step size determination is always a one-dimensional problem.
2. In unconstrained optimization, the slope of the cost function along the descent direction at zero step size is always positive.

3. The optimum step lies outside the interval of uncertainty.
 4. After initial bracketing, the golden section search requires two function evaluations to reduce the interval of uncertainty.
- 8.16 Find the minimum of the function $f(\alpha) = 7\alpha^2 - 20\alpha + 22$ using the equal interval search method within an accuracy of 0.001. Use $\delta = 0.05$.
 - 8.17 For the function $f(\alpha) = 7\alpha^2 - 20\alpha + 22$, use the golden section method to find the minimum with an accuracy of 0.005 (final interval of uncertainty should be less than 0.005). Use $\delta = 0.05$.
 - 8.18 Write a computer program to implement the alternate equal interval search process shown in Fig. 8.7 for any given function $f(\alpha)$. For the function $f(\alpha) = 2 - 4\alpha + e^\alpha$, use your program to find the minimum within an accuracy of 0.001. Use $\delta = 0.50$.
 - 8.19 Consider the function $f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$. Verify whether the vector $\mathbf{d} = (-12, -40, -48)$ at the point $(2, 4, 10)$ is a descent direction for f . What is the slope of the function at the given point? Find an optimum step size along \mathbf{d} by any numerical method.
 - 8.20 Consider the function $f(\mathbf{x}) = x_1^2 + x_2^2 - 2x_1 - 2x_2 + 4$. At the point $(1, 1)$, let a search direction be defined as $\mathbf{d} = (1, 2)$. Express f as a function of one variable at the given point along \mathbf{d} . Find an optimum step size along \mathbf{d} analytically.

For the following functions, direction of change at a point is given. Derive the function of one variable (line search function) that can be used to determine optimum step size (show all calculations).

- 8.21 $f(\mathbf{x}) = 0.1x_1^2 + x_2^2 - 10$; $\mathbf{d} = (-1, -2)$ at $\mathbf{x} = (5, 1)$
- 8.22 $f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2$; $\mathbf{d} = (-4, -6)$ at $\mathbf{x} = (4, 4)$
- 8.23 $f(\mathbf{x}) = 10(x_2 - x_1^2)^2 + (1 - x_1)^2$; $\mathbf{d} = (-162, 40)$ at $\mathbf{x} = (2, 2)$
- 8.24 $f(\mathbf{x}) = (x_1 - 2)^2 + x_2^2$; $\mathbf{d} = (2, -2)$ at $\mathbf{x} = (1, 1)$
- 8.25 $f(\mathbf{x}) = 0.5x_1^2 + x_2^2 - x_1x_2 - 7x_1 - 7x_2$; $\mathbf{d} = (7, 6)$ at $\mathbf{x} = (1, 1)$
- 8.26 $f(\mathbf{x}) = (x_1 + x_2)^2 + (x_2 + x_3)^2$; $\mathbf{d} = (-4, -8, -4)$ at $\mathbf{x} = (1, 1, 1)$
- 8.27 $f(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2$; $\mathbf{d} = (-2, -4, 2)$ at $\mathbf{x} = (1, 2, -1)$
- 8.28 $f(\mathbf{x}) = (x_1 + 3x_2 + x_3)^2 + 4(x_1 - x_2)^2$; $\mathbf{d} = (1, 3, 1)$ at $\mathbf{x} = (-1, -1, -1)$
- 8.29 $f(\mathbf{x}) = 9 - 8x_1 - 6x_2 - 4x_3 + 2x_1^2 + 2x_2^2 + x_3^2 + 2x_1x_2 + 2x_2x_3$; $\mathbf{d} = (2, -2, 0)$ at $\mathbf{x} = (1, 1, 1)$
- 8.30 $f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 + (x_4 - 4)^2$; $\mathbf{d} = (-2, 2, -2, 2)$ at $\mathbf{x} = (2, 1, 4, 3)$

For the following problems, calculate the initial interval of uncertainty for the equal interval search with $\delta = 0.05$ at the given point and the search direction.

- | | |
|--------------------|--------------------|
| 8.31 Exercise 8.21 | 8.32 Exercise 8.22 |
| 8.33 Exercise 8.23 | 8.34 Exercise 8.24 |
| 8.35 Exercise 8.25 | 8.36 Exercise 8.26 |
| 8.37 Exercise 8.27 | 8.38 Exercise 8.28 |
| 8.39 Exercise 8.29 | 8.40 Exercise 8.30 |

For the following problems, calculate the initial interval of uncertainty for the golden section search with $\delta = 0.05$ at the given point and the search direction; then complete two iterations of the Phase II of the method.

- 8.41 Exercise 8.21 8.42 Exercise 8.22
 8.43 Exercise 8.23 8.44 Exercise 8.24
 8.45 Exercise 8.25 8.46 Exercise 8.26
 8.47 Exercise 8.27 8.48 Exercise 8.28
 8.49 Exercise 8.29 8.50 Exercise 8.30

Section 8.3 Search Direction Determination: Steepest Descent Method

8.51 Answer True or False.

1. The steepest descent method is convergent.
2. The steepest descent method can converge to a local maximum point starting from a point where the gradient of the function is nonzero.
3. Steepest descent directions are orthogonal to each other.
4. Steepest descent direction is orthogonal to the cost surface.

For the following problems, complete two iterations of the steepest descent method starting from the given design point.

- 8.52 $f(x_1, x_2) = x_1^2 + 2x_2^2 - 4x_1 - 2x_1x_2$; starting design (1, 1)
 8.53 $f(x_1, x_2) = 12.096x_1^2 + 21.504x_2^2 - 1.7321x_1 - x_2$; starting design (1, 1)
 8.54 $f(x_1, x_2) = 6.983x_1^2 + 12.415x_2^2 - x_1$; starting design (2, 1)
 8.55 $f(x_1, x_2) = 12.096x_1^2 + 21.504x_2^2 - x_2$; starting design (1, 2)
 8.56 $f(x_1, x_2) = 25x_1^2 + 20x_2^2 - 2x_1 - x_2$; starting design (3, 1)
 8.57 $f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$; starting design (1, 1, 1)
 8.58 $f(x_1, x_2) = 8x_1^2 + 8x_2^2 - 80\sqrt{x_1^2 + x_2^2} - 20x_2 + 100$
 $- 80\sqrt{x_1^2 + x_2^2} + 20x_2 + 100 - 5x_1 - 5x_2$

Starting design (4, 6); the step size may be approximated or calculated using a computer program.

- 8.59 $f(x_1, x_2) = 9x_1^2 + 9x_2^2 - 100\sqrt{x_1^2 + x_2^2} - 20x_2 + 100$
 $- 64\sqrt{x_1^2 + x_2^2} + 16x_2 + 64 - 5x_1 - 41x_2$
 Starting design (5, 2); the step size may be approximated or calculated using a computer program.
 8.60 $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$; starting design (5, 2)
 8.61 $f(x_1, x_2, x_3, x_4) = (x_1 - 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$
 Let the starting design be (1, 2, 3, 4).
 8.62 Solve Exercises 8.52 to 8.61 using the computer program given in Appendix D for the steepest descent method.

8.63 Consider the following three functions:

$$f_1 = x_1^2 + x_2^2 + x_3^2; \quad f_2 = x_1^2 + 10x_2^2 + 100x_3^2; \quad f_3 = 100x_1^2 + x_2^2 + 0.1x_3^2$$

Minimize f_1 , f_2 , and f_3 using the program for the steepest descent method given in Appendix D. Choose the starting design to be (1, 1, 2) for all functions. What do you conclude from observing the performance of the method on the foregoing functions?

8.64 Calculate the gradient of the following functions at the given points by the forward, backward, and central difference approaches with a 1 percent change in the point and compare them with the exact gradient:

1. $f(\mathbf{x}) = 12.096x_1^2 + 21.504x_2^2 - 1.7321x_1 - x_2$ at (5, 6)

2. $f(\mathbf{x}) = 50(x_2 - x_1^2)^2 + (2 - x_1)^2$ at (1, 2)

3. $f(\mathbf{x}) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$ at (1, 2, 3)

8.65 Consider the following optimization problem

$$\text{maximize } \sum_{i=1}^n u_i \frac{\partial f}{\partial x_i} = (\mathbf{c} \cdot \mathbf{u})$$

$$\text{subject to the constraint } \sum_{i=1}^n u_i^2 = 1$$

Here $\mathbf{u} = (u_1, u_2, \dots, u_n)$ are components of a unit vector. Solve this optimization problem and show that the \mathbf{u} that maximizes the preceding objective function is indeed in the direction of the gradient \mathbf{c} .

Section 8.4 Search Direction Determination: Conjugate Gradient Method

8.66 *Answer True or False.*

1. The conjugate gradient method usually converges faster than the steepest descent method.
2. Conjugate directions are computed from gradients of the cost function.
3. Conjugate directions are normal to each other.
4. The conjugate direction at the k th point is orthogonal to the gradient of the cost function at the $(k + 1)$ th point when an exact step size is calculated.
5. The conjugate direction at the k th point is orthogonal to the gradient of the cost function at the $(k - 1)$ th point.

For the following problems, complete two iterations of the conjugate gradient method.

8.67 Exercise 8.52 8.68 Exercise 8.53

8.69 Exercise 8.54 8.70 Exercise 8.55

8.71 Exercise 8.56 8.72 Exercise 8.57

8.73 Exercise 8.58 8.74 Exercise 8.59

8.75 Exercise 8.60 8.76 Exercise 8.61

8.77 Write a computer program to implement the conjugate gradient method (or, modify the steepest descent program given in Appendix D). Solve Exercises 8.52 to 8.61 using your program.

For the following problems, write an Excel worksheet and solve the problems using Solver.

8.78 Exercise 8.52 8.79 Exercise 8.53

8.80 Exercise 8.54 8.81 Exercise 8.55

8.82 Exercise 8.56 8.83 Exercise 8.57

8.84 Exercise 8.58 8.85 Exercise 8.59

8.86 Exercise 8.60 8.87 Exercise 8.61

9 More on Numerical Methods for Unconstrained Optimum Design

Upon completion of this chapter, you will be able to:

- Use some alternate procedures for step size calculation
- Explain properties of the gradient vector used in the steepest descent method
- Use scaling of design variables to improve performance of optimization methods
- Use the second-order methods for unconstrained optimization, such as the Newton method and understand its limitations
- Use approximate second-order methods for unconstrained optimization, called quasi-Newton methods
- Transform constrained problems to unconstrained problems and use unconstrained optimization methods to solve them

The material of this chapter builds upon the basic concepts and numerical methods for unconstrained problems presented in the previous chapter. Topics covered include polynomial interpolation for step size calculation, properties of the gradient vector, a Newton method that uses Hessian of the cost function in numerical optimization, scaling of design variables, approximate second-order methods—called quasi-Newton methods, and transformation methods that transform a constrained problem to an unconstrained problem so that unconstrained optimization methods can be used to solve constrained problems. These topics may be omitted in an undergraduate course on optimum design or on first independent reading of the text.

9.1 More on Step Size Determination

The interval reducing methods described in Chapter 8 can require too many function evaluations during line search to determine an appropriate step size. In realistic engineering design problems, the function evaluation requires a significant amount of computational effort. Therefore, methods such as golden section search are inefficient for many practical applications. In this section, we present some other line search methods such as polynomial interpolation and inaccurate line search.

9.1.1 Polynomial Interpolation

Instead of evaluating the function at numerous trial points, we can pass a curve through a limited number of points and use the analytical procedure to calculate the step size. Any continuous function on a given interval can be approximated as closely as desired by passing a higher order polynomial through its data points and then calculating its minimum explicitly. The minimum point of the approximating polynomial is often a good estimate of the exact minimum of the line search function $f(\alpha)$. Thus, polynomial interpolation can be an efficient technique for one-dimensional search. Whereas many polynomial interpolation schemes can be devised, we will present two procedures based on quadratic interpolation.

Quadratic Curve Fitting Many times it is sufficient to approximate the function $f(\alpha)$ on an interval of uncertainty by a quadratic function. To replace a function in an interval with a quadratic function, we need to know the function value at three distinct points to determine the three coefficients of the quadratic polynomial. It must also be assumed that the function $f(\alpha)$ is sufficiently smooth and unimodal, and that the initial interval of uncertainty (α_l, α_u) is known. Let α_i be any intermediate point in the interval (α_l, α_u) , and let $f(\alpha_l)$, $f(\alpha_i)$, and $f(\alpha_u)$ be the function values at the respective points. Figure 9-1 shows the function $f(\alpha)$ and the quadratic function $q(\alpha)$ as its approximation in the interval (α_l, α_u) . $\bar{\alpha}$ is the minimum point of the quadratic function $q(\alpha)$ whereas α^* is the exact minimum point of $f(\alpha)$. An iteration can be used to improve the estimate $\bar{\alpha}$ for α^* .

Any quadratic function $q(\alpha)$ can be expressed in the general form as

$$q(\alpha) = a_0 + a_1\alpha + a_2\alpha^2 \quad (9.1)$$

where a_0 , a_1 , and a_2 are the unknown coefficients. Since the function $q(\alpha)$ must have the same value as the function $f(\alpha)$ at the points α_l , α_i , and α_u , we get three equations in three unknowns a_0 , a_1 , and a_2 as follows:

$$a_0 + a_1\alpha_l + a_2\alpha_l^2 = f(\alpha_l)$$

$$a_0 + a_1\alpha_i + a_2\alpha_i^2 = f(\alpha_i)$$

$$a_0 + a_1\alpha_u + a_2\alpha_u^2 = f(\alpha_u)$$

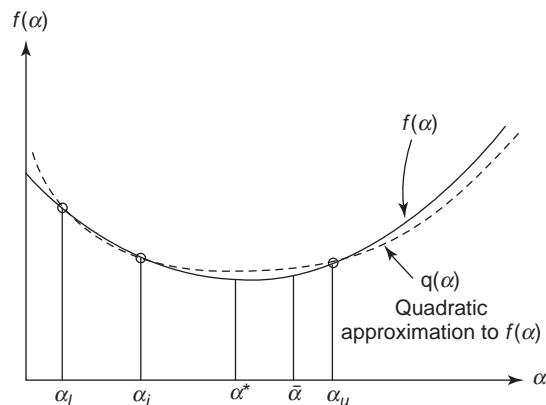


FIGURE 9-1 Quadratic approximation for a function $f(\alpha)$.

Solving the system of linear simultaneous equations for a_0 , a_1 , and a_2 , we get:

$$a_2 = \frac{1}{(\alpha_u - \alpha_i)} \left[\frac{f(\alpha_u) - f(\alpha_i)}{(\alpha_u - \alpha_i)} - \frac{f(\alpha_i) - f(\alpha_l)}{(\alpha_i - \alpha_l)} \right]$$

$$a_1 = \frac{f(\alpha_i) - f(\alpha_l)}{(\alpha_i - \alpha_l)} - a_2(\alpha_i + \alpha_l) \quad (9.2)$$

$$a_0 = f(\alpha_l) - a_1\alpha_l - a_2\alpha_l^2$$

The minimum point $\bar{\alpha}$ of the quadratic curve $q(\alpha)$ in Eq. (9.1) is calculated by solving the necessary condition $dq/d\alpha = 0$ and verifying the sufficiency condition $d^2q/d\alpha^2 > 0$, as

$$\bar{\alpha} = -\frac{1}{2a_2} a_1; \quad \text{if } \frac{d^2q}{d\alpha^2} = 2a_2 > 0 \quad (9.3)$$

Thus, if $a_2 > 0$, $\bar{\alpha}$ is a minimum of $q(\alpha)$. Additional iterations may be used to further refine the interval of uncertainty. The quadratic curve fitting technique may now be given in the form of a computational algorithm:

- Step 1.* Select a small number δ , and locate the initial interval of uncertainty (α_l, α_u) . Any zero-order method discussed previously may be used.
- Step 2.* Let α_i be an intermediate point in the interval (α_l, α_u) and $f(\alpha_i)$ be the value of $f(\alpha)$ at α_i .
- Step 3.* Compute the coefficients a_0 , a_1 , and a_2 from Eqs. (9.2), $\bar{\alpha}$ from Eq. (9.3), and $f(\bar{\alpha})$.
- Step 4.* Compare α_i and $\bar{\alpha}$. If $\alpha_i < \bar{\alpha}$, continue with this step. Otherwise, go to Step 5.
- (a) If $f(\alpha_i) < f(\bar{\alpha})$, then $\alpha_l \leq \alpha^* \leq \bar{\alpha}$. The new limits of the reduced interval of uncertainty are $\alpha'_l = \alpha_l$, $\alpha'_u = \bar{\alpha}$, and $\alpha'_i = \alpha_i$ and go to Step 6.
- (b) If $f(\alpha_i) > f(\bar{\alpha})$, then $\bar{\alpha} \leq \alpha^* \leq \alpha_u$. The new limits of the reduced interval of uncertainty are $\alpha'_l = \bar{\alpha}$, $\alpha'_u = \alpha_u$, and $\alpha'_i = \alpha_i$ and go to Step 6.
- Step 5.* (a) If $f(\alpha_i) < f(\bar{\alpha})$, then $\alpha_l \leq \alpha^* \leq \alpha_i$. The new limits of the reduced interval of uncertainty are $\alpha'_l = \bar{\alpha}$, $\alpha'_u = \alpha_u$, and $\alpha'_i = \alpha_i$ and go to Step 6.
- (b) If $f(\alpha_i) > f(\bar{\alpha})$, then $\alpha_l \leq \alpha^* \leq \alpha_i$. The new limits for the reduced interval of uncertainty are $\alpha'_l = \alpha_l$, $\alpha'_u = \alpha_i$, and $\alpha'_i = \bar{\alpha}$ and go to Step 6.
- Step 6.* If the two successive estimates of the minimum point of $f(\alpha)$ are sufficiently close, then stop. Otherwise, delete the primes on α'_l , α'_i , and α'_u and return to Step 2.

Example 9.1 illustrates evaluation of the step size using quadratic interpolation.

EXAMPLE 9.1 One-dimensional Minimization with Quadratic Interpolation

Find the minimum point of $f(\alpha) = 2 - 4\alpha + e^\alpha$ of Example 8.3 by polynomial interpolation. Use the golden section search with $\delta = 0.5$ to bracket the minimum point initially.

Solution.

Iteration 1. From Example 8.3 the following information is known.

$$\begin{aligned}\alpha_l &= 1.2077, & \alpha_i &= 1.309017, & \alpha_u &= 2.618034 \\ f(\alpha_l) &= 1.648721, & f(\alpha_i) &= 0.466464, & f(\alpha_u) &= 5.236610\end{aligned}$$

The coefficients a_0 , a_1 , and a_2 are calculated from Eq. (9.2) as

$$a_2 = \frac{1}{1.30902} \left(\frac{3.5879}{2.1180} - \frac{-1.1823}{0.80902} \right) = 2.410$$

$$a_1 = \frac{-1.1823}{0.80902} - (2.41)(1.80902) = -5.821$$

$$a_0 = 1.648271 - (-5.821)(0.50) - 2.41(0.25) = 3.957$$

Therefore, $\bar{\alpha} = 1.2077$ from Eq. (9.3), and $f(\bar{\alpha}) = 0.5149$. Note that $\bar{\alpha} < \alpha_i$ and $f(\alpha_i) < f(\bar{\alpha})$. Thus, new limits of the reduced interval of uncertainty are $\alpha'_l = \bar{\alpha} = 1.2077$, $\alpha'_u = \alpha_u = 2.618034$, and $\alpha'_i = \alpha_i = 1.309017$.

Iteration 2. We have the new limits for the interval of uncertainty, the intermediate point, and the respective values as

$$\begin{aligned}\alpha_l &= 1.2077, & \alpha_i &= 1.309017, & \alpha_u &= 2.618034 \\ f(\alpha_l) &= 0.5149, & f(\alpha_i) &= 0.466464, & f(\alpha_u) &= 5.23661\end{aligned}$$

The coefficients a_0 , a_1 , and a_2 are calculated as before, $a_0 = 5.7129$, $a_1 = -7.8339$, and $a_2 = 2.9228$. Thus, $\bar{\alpha} = 1.34014$ and $f(\bar{\alpha}) = 0.4590$.

Comparing these results with the optimum solution given in Table 8-1, we observe that $\bar{\alpha}$ and $f(\bar{\alpha})$ are quite close to the final solution. One more iteration can give a very good approximation to the optimum step size. Note that only five function evaluations are used to obtain a fairly accurate optimum step size for the function $f(\alpha)$. Therefore, the polynomial interpolation approach can be quite efficient for one-dimensional minimization.

Alternate Quadratic Interpolation In this approach, we use the known information about the function at $\alpha = 0$ to perform quadratic interpolation; i.e., we can use $f(0)$ and $f'(0)$ in the interpolation process. Example 9.2 illustrates this alternate quadratic interpolation procedure.

EXAMPLE 9.2 One-Dimensional Minimization with Alternate Quadratic Interpolation

Find the minimum point of $f(\alpha) = 2 - 4\alpha + e^\alpha$ using $f(0)$, $f'(0)$, and $f(\alpha_u)$ to fit a quadratic curve, where α_u is an upper bound on the minimum point of $f(\alpha)$.

Solution. Let the general equation for a quadratic curve be $a_0 + a_1\alpha + a_2\alpha^2$, where a_0 , a_1 , and a_2 are the unknown coefficients. Let us select the upper bound on α^* to be

2.618034 (α_n) from the golden section search. Using the given function $f(\alpha)$, we have $f(0) = 3$, $f(2.618034) = 5.23661$, and $f'(0) = -3$. Now, as before, we get the following three equations to solve for the unknown coefficients a_0 , a_1 , and a_2 :

$$a_0 = f(0) = 3$$

$$a_0 + 2.618034a_1 + 6.854a_2 = f(2.618034) = 5.23661$$

$$a_1 = f'(0) = -3$$

Solving the three equations simultaneously, we get $a_0 = 3$, $a_1 = -3$, and $a_2 = 1.4722$. The minimum point of the parabolic curve using Eq. (9.3) is given as $\bar{\alpha} = 1.0189$ and $f(\bar{\alpha}) = 0.69443$. This estimate can be improved using an iteration as demonstrated in Example 9.1. Note that an estimate of the minimum point of the function $f(\alpha)$ can be found in only two function evaluations. Since the slope $f'(0) = \mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)}$ is known for multidimensional problems, no additional calculations are required to evaluate it at $\alpha = 0$.

9.1.2 Inaccurate Line Search

Exact line search during unconstrained or constrained minimization can be quite time consuming. Therefore, usually, the inaccurate line search procedures that also satisfy global convergence requirements are used in most computer implementations. The *basic concept* of inaccurate line search is that the step size should not be too small or too large, and there should be sufficient decrease in the cost function value. Several inaccurate line search procedures have been developed and used. Here, we discuss some basic concepts and present a procedure for inaccurate line search.

Recall that a step size $\alpha_k > 0$ exists if $\mathbf{d}^{(k)}$ satisfies the *descent condition* $(\mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)}) < 0$. Generally, an iterative method, such as quadratic interpolation, is used during line search, and the process is terminated when the step size is sufficiently accurate; i.e., the line search termination criterion $(\mathbf{c}^{(k+1)} \cdot \mathbf{d}^{(k)}) = 0$ of Eq. (8.11) is satisfied sufficiently accurately. However, note that to check this condition, we need to calculate the gradient of the cost function at each trial step size, which can be quite expensive. Therefore, some other simple strategies have been developed that do not require this calculation. One such strategy is called the *Armijo's rule*. The essential idea is first to guarantee that the selected step size α is not too large; i.e., the current step is not far beyond the optimum step size. Next, the step size should not be too small such that there is little progress toward the minimum point (i.e., there is very little reduction in the cost function).

Let the line search function be defined as $f(\alpha) = f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$ as in Eq. (8.9). Armijo's rule uses a linear function of α as $f(0) + \alpha[\rho f'(0)]$, where ρ is a fixed number between 0 and 1; $0 < \rho < 1$. This function is shown as the dashed line in Fig. 9-2. A value of α is considered *not too large* if the corresponding function value lies below the dashed line, i.e.,

$$f(\alpha) \leq f(0) + \alpha[\rho f'(0)] \quad (9.4)$$

To ensure that α is *not too small*, a number $\eta > 1$ is selected. Then α is considered not too small if it satisfies the following inequality:

$$f(\eta\alpha) > f(0) + \eta\alpha[\rho f'(0)] \quad (9.5)$$

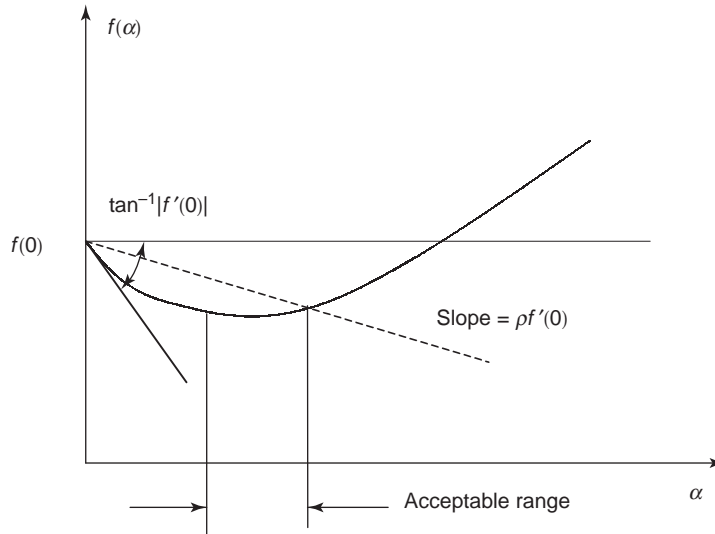


FIGURE 9-2 Inaccurate line search using Armijo's rule.

This means that if α is increased by a factor η , it will not meet the test given in Eq. (9.4).

Armijo's rule can be used to *determine* the step size without interpolation as follows: one starts with an arbitrary α . If it satisfies Eq. (9.4), it is repeatedly increased by η ($\eta = 2$ to 10 and $\rho = 0.2$ are often used) until Eq. (9.4) is violated. The largest α satisfying Eq. (9.4) is selected as the step size. If on the other hand, the starting value of α does not satisfy Eq. (9.4), it is repeatedly divided by η until Inequality (9.4) is satisfied. Use of a procedure similar to Armijo's rule is demonstrated in a numerical algorithm for constrained problems in Chapter 11.

9.2 More on Steepest Descent Method

In this section we shall study properties of the gradient vector that is used in the steepest descent method. Proofs of the properties are given since they are quite instructive. We shall also show that the steepest descent directions at successive iterations are orthogonal to each other.

9.2.1 Properties of the Gradient Vector

Property 1 The gradient vector \mathbf{c} of a function $f(x_1, x_2, \dots, x_n)$ at the point $\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_n^*)$ is orthogonal (normal) to the tangent hyperplane for the surface $f(x_1, x_2, \dots, x_n) = \text{constant}$.

This is an important property of the gradient vector shown graphically in Fig. 9-3. It shows the surface $f(\mathbf{x}) = \text{constant}$; \mathbf{x}^* is a point on the surface; C is any curve on the surface through the point \mathbf{x}^* ; \mathbf{T} is a vector tangent to the curve C at the point \mathbf{x}^* ; \mathbf{u} is any unit vector; and \mathbf{c} is the gradient vector at \mathbf{x}^* . According to the above property, vectors \mathbf{c} and \mathbf{T} are normal to each other, i.e., their dot product is zero, $\mathbf{c} \cdot \mathbf{T} = 0$.

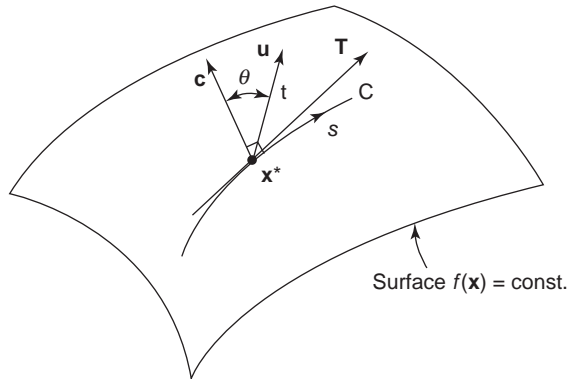


FIGURE 9-3 Gradient vector for the surface $f(\mathbf{x}) = \text{constant}$ at the point \mathbf{x}^* .

Proof To show this, we take any curve C on the surface $f(x_1, x_2, \dots, x_n) = \text{constant}$, as shown in Fig. 9-3. Let the curve pass through the point $\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_n^*)$. Also, let s be a parameter along C . Then a unit tangent vector \mathbf{T} along C at the point \mathbf{x}^* is given as

$$\mathbf{T} = \left[\frac{\partial x_1}{\partial s} \quad \frac{\partial x_2}{\partial s} \quad \dots \quad \frac{\partial x_n}{\partial s} \right]^T \quad (\text{a})$$

Since $f(\mathbf{x}) = \text{constant}$, the derivative of f along the curve C is zero, i.e., $df/ds = 0$ (directional derivative of f in the direction s). Or, using the chain rule of differentiation, we get

$$\frac{df}{ds} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial s} + \dots + \frac{\partial f}{\partial x_n} \frac{\partial x_n}{\partial s} = 0 \quad (\text{b})$$

Writing Eq. (b) in the vector form after identifying $\partial f/\partial x_i$ and $\partial x_i/\partial s$ [from Eq. (a)] as components of the gradient and the unit tangent vectors, we obtain $\mathbf{c} \cdot \mathbf{T} = 0$, or $\mathbf{c}^T \mathbf{T} = 0$. Since the dot product of the gradient vector \mathbf{c} with the tangential vector \mathbf{T} is zero, the vectors are normal to each other. But, \mathbf{T} is any tangent vector at \mathbf{x}^* , and so \mathbf{c} is orthogonal to the tangent plane for the surface $f(\mathbf{x}) = \text{constant}$ at the point \mathbf{x}^* .

Property 2 The second property is that the gradient represents a direction of maximum rate of increase for the function $f(\mathbf{x})$ at the point \mathbf{x}^* .

Proof To show this, let \mathbf{u} be a unit vector in any direction that is not tangent to the surface. This is shown in Fig. 9-3. Let t be a parameter along \mathbf{u} . The derivative of $f(\mathbf{x})$ in the direction \mathbf{u} at the point \mathbf{x}^* (i.e., directional derivative of f) is given as

$$\frac{df}{dt} = \lim_{\varepsilon \rightarrow 0} \frac{f(\mathbf{x} + \varepsilon \mathbf{u}) - f(\mathbf{x})}{\varepsilon} \quad (\text{c})$$

where ε is a small number. Using Taylor's expansion, we have

$$f(\mathbf{x} + \varepsilon \mathbf{u}) = f(\mathbf{x}) + \varepsilon \left[u_1 \frac{\partial f}{\partial x_1} + u_2 \frac{\partial f}{\partial x_2} + \dots + u_n \frac{\partial f}{\partial x_n} \right] + o(\varepsilon^2)$$

where u_i are components of the unit vector \mathbf{u} and $o(\varepsilon^2)$ are terms of order ε^2 . Rewriting the foregoing equation,

$$f(\mathbf{x} + \varepsilon\mathbf{u}) - f(\mathbf{x}) = \varepsilon \sum_{i=1}^n u_i \frac{\partial f}{\partial x_i} + o(\varepsilon^2) \quad (d)$$

Substituting Eq. (d) into Eq. (c) and taking the indicated limit, we get

$$\frac{df}{dt} = \varepsilon \sum_{i=1}^n u_i \frac{\partial f}{\partial x_i} = \mathbf{c} \cdot \mathbf{u} = \mathbf{c}^T \mathbf{u} \quad (e)$$

Using the definition of the dot product in Eq. (e), we get

$$\frac{df}{dt} = \|\mathbf{c}\| \|\mathbf{u}\| \cos \theta \quad (f)$$

where θ is the angle between the \mathbf{c} and \mathbf{u} vectors. The right side of Eq. (f) will have extreme values when $\theta = 0^\circ$ or 180° . When $\theta = 0^\circ$, vector \mathbf{u} is along \mathbf{c} and $\cos \theta = 1$. Therefore, from Eq. (f), df/dt represents the maximum rate of increase for $f(\mathbf{x})$ when $\theta = 0^\circ$. Similarly, when $\theta = 180^\circ$, vector \mathbf{u} points in the negative \mathbf{c} direction. Therefore, from Eq. (f), df/dt represents the maximum rate of decrease for $f(\mathbf{x})$ when $\theta = 180^\circ$.

According to the foregoing property of the gradient vector, if we need to move away from the surface $f(\mathbf{x}) = \text{constant}$, the function increases most rapidly along the gradient vector compared with a move in any other direction. In Fig. 9-3, a small move along the direction \mathbf{c} will result in a larger increase in the function, compared with a similar move along the direction \mathbf{u} . Of course, any small move along the direction \mathbf{T} results in no change in the function since \mathbf{T} is tangent to the surface.

Property 3 The maximum rate of change of $f(\mathbf{x})$ at any point \mathbf{x}^* is the magnitude of the gradient vector.

Proof Since \mathbf{u} is a unit vector, the maximum value of df/dt from Eq. (f) is given as

$$\max \left| \frac{df}{dt} \right| = \|\mathbf{c}\|$$

However, for $\theta = 0^\circ$, \mathbf{u} is in the direction of the gradient vector. Therefore, the magnitude of the gradient represents the maximum rate of change for the function $f(\mathbf{x})$.

These properties show that the gradient vector at any point \mathbf{x}^* represents a direction of maximum increase in the function $f(\mathbf{x})$ and the rate of increase is the magnitude of the vector. The gradient is therefore called a direction of steepest ascent for the function $f(\mathbf{x})$. Example 9.3 verifies properties of the gradient vector.

EXAMPLE 9.3 Verification of Properties of the Gradient Vector

Verify the properties of the gradient vector for the function $f(\mathbf{x}) = 25x_1^2 + x_2^2$ at the point $\mathbf{x}^{(0)} = (0.6, 4)$.

Solution. Figure 9-4 shows in the $x_1 - x_2$ plane the contours of value 25 and 100 for the function f . The value of the function at $(0.6, 4)$ is $f(0.6, 4) = 25$. The gradient of the function at $(0.6, 4)$ is given as

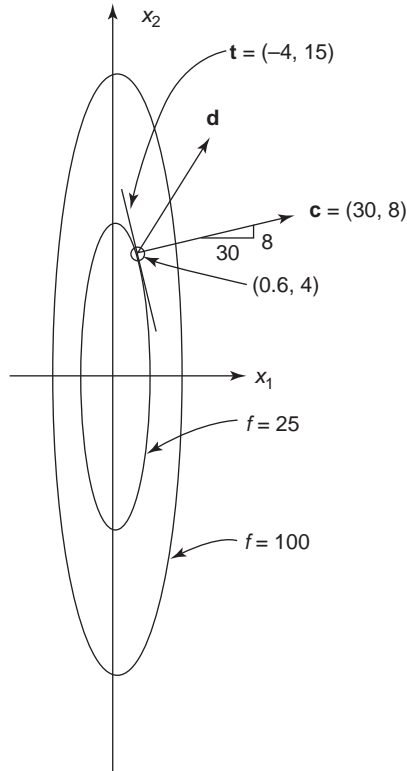


FIGURE 9-4 Contours of function $f = 25x_1^2 + x_2^2$ for $f = 25$ and 100 .

$$\mathbf{c} = \nabla f(0.6, 4) = (\partial f / \partial x_1, \partial f / \partial x_2) = (50x_1, 2x_2) = (30, 8) \quad (\text{a})$$

$$\|\mathbf{c}\| = \sqrt{30 \times 30 + 8 \times 8} = 31.04835 \quad (\text{b})$$

Therefore, a unit vector along the gradient is given as

$$\mathbf{C} = \mathbf{c} / \|\mathbf{c}\| = (0.966235, 0.257663) \quad (\text{c})$$

Using the given function, a vector tangent to the curve at the point $(0.6, 4)$ is given as

$$\mathbf{t} = (-4, 15) \quad (\text{d})$$

This vector is obtained by differentiating the equation for the curve $25x_1^2 + x_2^2 = 25$ at the point $(0.6, 4)$ with respect to the parameter s along the curve. This gives the expression $\partial x_1 / \partial s = -(4/15) \partial x_2 / \partial s$. Then the vector \mathbf{t} tangent to the curve is obtained using Eq. (a) as $(\partial x_1 / \partial s, \partial x_2 / \partial s)$. The unit tangent vector is calculated as

$$\mathbf{T} = \mathbf{t} / \|\mathbf{t}\| = (-0.257663, 0.966235) \quad (\text{e})$$

Property 1. If the gradient is normal to the tangent, then $\mathbf{C} \cdot \mathbf{T} = 0$. This is indeed true for the preceding data. We can also use the condition that if two lines are orthogonal, then $m_1 m_2 = -1$, where m_1 and m_2 are the slopes of the two lines (this result can

be proved by using the rotational transformation of coordinates through 90 degrees). To calculate the slope of the tangent, we use the equation for the curve $25x_1^2 + x_2^2 = 25$, or $x_2 = 5\sqrt{1-x_1^2}$. Therefore, the slope of the tangent at the point (0.6, 4) is given as

$$m_1 = \frac{dx_2}{dx_1} = -5x_1/\sqrt{1-x_1^2} = -\frac{15}{4} \quad (f)$$

This slope is also obtained directly from the tangent vector $\mathbf{t} = (-4, 15)$. The slope of the gradient vector $\mathbf{c} = (30, 8)$ is $m_2 = \frac{8}{30} = \frac{4}{15}$. Thus $m_1 m_2$ is, indeed, -1 , and the two lines are normal to each other.

Property 2. Consider any arbitrary direction $\mathbf{d} = (0.501034, 0.865430)$ at the point (0.6, 4) as shown in Fig. 9-4. If \mathbf{C} is the direction of steepest ascent, then the function should increase more rapidly along \mathbf{C} than along \mathbf{d} . Let us choose a step size $\alpha = 0.1$ and calculate two points, one along \mathbf{C} and the other along \mathbf{d} as

$$\begin{aligned} \mathbf{x}^{(1)} &= \mathbf{x}^{(0)} + \alpha \mathbf{C} \\ &= \begin{bmatrix} 0.6 \\ 4.0 \end{bmatrix} + 0.1 \begin{bmatrix} 0.966235 \\ 0.257633 \end{bmatrix} = \begin{bmatrix} 0.6966235 \\ 4.0257663 \end{bmatrix} \end{aligned} \quad (g)$$

$$\begin{aligned} \mathbf{x}^{(2)} &= \mathbf{x}^{(0)} + \alpha \mathbf{d} \\ &= \begin{bmatrix} 0.6 \\ 4.0 \end{bmatrix} + 0.1 \begin{bmatrix} 0.501034 \\ 0.865430 \end{bmatrix} = \begin{bmatrix} 0.6501034 \\ 4.0865430 \end{bmatrix} \end{aligned} \quad (h)$$

Now, we calculate the function at these points and compare their values: $f(\mathbf{x}^{(1)}) = 28.3389$, $f(\mathbf{x}^{(2)}) = 27.2657$. Since $f(\mathbf{x}^{(1)}) > f(\mathbf{x}^{(2)})$, the function increases more rapidly along \mathbf{C} than along \mathbf{d} .

Property 3. If the magnitude of the gradient vector represents the maximum rate of change of $f(\mathbf{x})$, then $(\mathbf{c} \cdot \mathbf{c}) > (\mathbf{c} \cdot \mathbf{d})$, $(\mathbf{c} \cdot \mathbf{c}) = 964.0$, and $(\mathbf{c} \cdot \mathbf{d}) = 21.9545$. Therefore, the gradient vector satisfies this property also.

Note that the last two properties are valid only in a local sense, i.e., only in a small neighborhood of the point at which the gradient is evaluated.

9.2.2 Orthogonality of Steepest Descent Directions

It is interesting to note that the successive directions of steepest descent are normal to one another, i.e., $(\mathbf{c}^{(k)} \cdot \mathbf{c}^{(k+1)}) = 0$. This can be shown quite easily by using the necessary conditions to determine the optimum step size. The step size determination problem is to compute α_k that minimizes $f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$. The necessary condition for this is $df/d\alpha = 0$. Using the chain rule of differentiation, we get

$$\frac{df(\mathbf{x}^{(k+1)})}{d\alpha} = \left[\frac{\partial f(\mathbf{x}^{(k+1)})}{\partial \mathbf{x}} \right]^T \frac{\partial \mathbf{x}^{(k+1)}}{\partial \alpha} \quad (9.6a)$$

which gives

$$(\mathbf{c}^{(k+1)} \cdot \mathbf{d}^{(k)}) = 0 \quad \text{or} \quad (\mathbf{c}^{(k+1)} \cdot \mathbf{c}^{(k)}) = 0 \quad (9.6b)$$

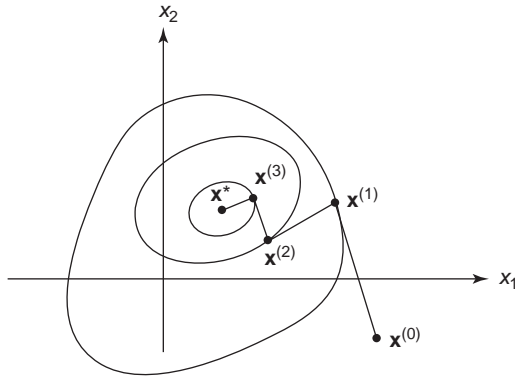


FIGURE 9-5 Orthogonal steepest descent paths.

$$\mathbf{c}^{(k+1)} = \frac{\partial f(\mathbf{x}^{(k+1)})}{\partial \mathbf{x}} \quad \text{and} \quad \frac{\partial \mathbf{x}^{(k+1)}}{\partial \alpha} = \frac{\partial}{\partial \alpha} (\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}) = \mathbf{d}^{(k)} \quad (9.6c)$$

In the two-dimensional case, $\mathbf{x} = (x_1, x_2)$. Figure 9-5 is a view of the design variable space. The closed curves in the figure are contours of the cost function $f(\mathbf{x})$. The figure shows several steepest descent directions that are orthogonal to each other.

9.3 Scaling of Design Variables

The rate of convergence of the steepest descent method is at best linear even for a quadratic cost function. It is possible to accelerate this rate of convergence of the steepest descent method if the condition number of the Hessian of the cost function can be reduced by scaling the design variables. For a quadratic cost function it is possible to scale the design variables such that the condition number of the Hessian matrix with respect to the new design variables, is unity (the *condition number* of a matrix is calculated as the ratio of the largest to the smallest eigenvalues of the matrix). The steepest descent method converges in only one iteration for a positive definite quadratic function with a unit condition number. To obtain the optimum point with the original design variables, we could then unscale the transformed design variables. Thus the main objective of scaling the design variables is to define transformations such that the condition number of the Hessian with respect to the transformed variables is 1. We shall demonstrate the advantage of scaling the design variables with Examples 9.4 and 9.5.

EXAMPLE 9.4 Effect of Scaling of Design Variables

Minimize $f(x_1, x_2) = 25x_1^2 + x_2^2$ with a starting design $(1, 1)$ by the steepest descent method. How would you scale the design variables to accelerate the rate of convergence?

Solution. Let us solve the problem by the computer program for the steepest descent method given in Appendix D. The results are summarized in Table 9-1. Note the inefficiency of the method on such a simple quadratic cost function; the method takes 5 iterations and 111 function evaluations. Figure 9-6 shows the contours of the cost function and the progress of the method from the initial design.

TABLE 9-1 Optimum Solution for Example 9.4 with Steepest Descent Method: $f(x) = 25x_1^2 + x_2^2$

Starting values of design variables:	1, 1
Optimum design variables:	-2.35450E-06, 1.37529E-03
Optimum cost function value:	1.89157E-06
Norm of gradient at optimum:	2.75310E-03
Number of iterations:	5
Number of function evaluations:	111

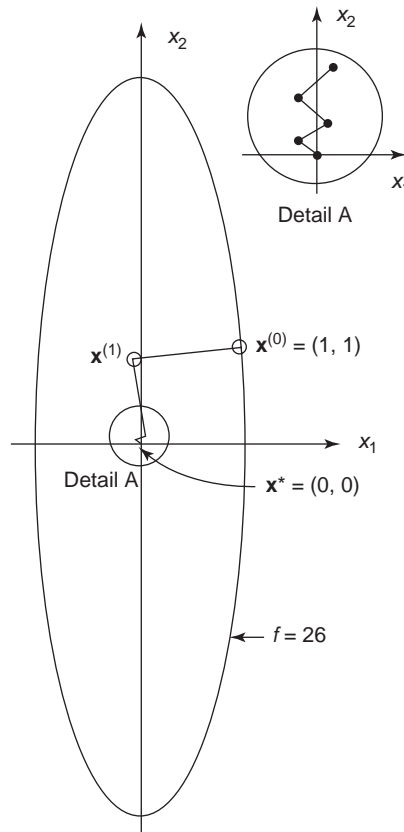


FIGURE 9-6 Iteration history for Example 9.4 with the steepest descent method.

The Hessian of $f(x_1, x_2)$ is a diagonal matrix given as

$$\mathbf{H} = \begin{bmatrix} 50 & 0 \\ 0 & 2 \end{bmatrix}$$

The condition number of the Hessian is $50/2 = 25$ since its eigenvalues are 50 and 2. Now let us introduce new design variables y_1 and y_2 such that

$$\mathbf{x} = \mathbf{D}\mathbf{y} \quad \text{where} \quad \mathbf{D} = \begin{bmatrix} \frac{1}{\sqrt{50}} & 0 \\ 0 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Note that, in general, we may use $D_{ii} = 1/\sqrt{H_{ii}}$ for $i = 1$ to n if the Hessian is a diagonal matrix (the diagonal elements are the eigenvalues of \mathbf{H}). The previous transformation gives $x_1 = y_1/\sqrt{50}$ and $x_2 = y_2/\sqrt{2}$ and $f(y_1, y_2) = \frac{1}{2}(y_1^2 + y_2^2)$. The minimum point of $f(y_1, y_2)$ is found in just one iteration by the steepest descent method compared with the five iterations for the original function since the condition number of the transformed Hessian is 1. The optimum point is (0, 0) in the new design variable space. To obtain the minimum point in the original design space, we have to unscale the transformed design variables as $x_1^* = y_1/\sqrt{50} = 0$ and $x_2^* = y_2/\sqrt{2} = 0$. Thus for this example, the use of design variable scaling is quite beneficial.

EXAMPLE 9.5 Effect of Scaling of Design Variables

Minimize $f(x_1, x_2) = 6x_1^2 - 6x_1x_2 + 2x_2^2 - 5x_1 + 4x_2 + 2$ (a)

with a starting design $(-1, -2)$ by the steepest descent method. Scale the design variables to have a condition number of unity for the Hessian matrix of the function with respect to the new design variables.

Solution. Note that unlike the previous example the function f in this problem contains the cross term x_1x_2 . Therefore the Hessian matrix is not a diagonal matrix, and we need to compute its eigenvalues and eigenvectors to find a suitable scaling or transformation of the design variables. The Hessian \mathbf{H} of the function f is given as

$$\mathbf{H} = \begin{bmatrix} 12 & -6 \\ -6 & 4 \end{bmatrix} \quad (b)$$

The eigenvalues of the Hessian are calculated as 0.7889 and 15.211 (condition number = $15.211/0.7889 = 19.3$). The corresponding eigenvectors are (0.4718, 0.8817) and $(-0.8817, 0.4718)$. Now let us define new variables y_1 and y_2 by the following transformation

$$\mathbf{x} = \mathbf{Q}\mathbf{y} \quad \text{where} \quad \mathbf{Q} = \begin{bmatrix} 0.4718 & -0.8817 \\ 0.8817 & 0.4718 \end{bmatrix} \quad (c)$$

Note that the columns of \mathbf{Q} are the eigenvectors of the Hessian matrix \mathbf{H} . The transformation of variables defined by Eq. (c) gives the function in terms of y_1 and y_2 as

$$f(y_1, y_2) = 0.5(0.7889y_1^2 + 15.211y_2^2) + 1.678y_1 + 6.2957y_2 + 2 \quad (d)$$

The condition number of the Hessian matrix in the new design variables y_1 and y_2 is still not unity. To achieve the condition number equal to unity for the Hessian, we must define another transformation of y_1 and y_2 using the eigenvalues of the Hessian matrix as

$$\mathbf{y} = \mathbf{D}\mathbf{z}, \quad \text{where} \quad \mathbf{D} = \begin{bmatrix} \frac{1}{\sqrt{0.7889}} & 0 \\ 0 & \frac{1}{\sqrt{15.211}} \end{bmatrix} \quad (e)$$

where z_1 and z_2 are the new design variables that can be calculated from the equations:

$$y_1 = \frac{z_1}{\sqrt{0.7889}} \quad \text{and} \quad y_2 = \frac{z_2}{\sqrt{15.211}} \quad (f)$$

and $f(z_1, z_2) = 0.5(z_1^2 + z_2^2) + 1.3148z_1 + 1.6142z_2$. Note that the condition number of the Hessian of $f(z_1, z_2)$ is 1. The steepest descent method converges to the solution of $f(z_1, z_2)$ in just one iteration as $(-1.3158, -1.6142)$. The minimum point in the original design space is found by defining the inverse transformation as $\mathbf{x} = \mathbf{QDz}$. This gives the minimum point in the original design space as $(-\frac{1}{3}, -\frac{3}{2})$.

9.4 Search Direction Determination: Newton's Method

With the steepest descent method, only first-order derivative information is used to determine the search direction. If second-order derivatives were available, we could use them to represent the cost surface more accurately, and a better search direction could be found. With the inclusion of second-order information, we could expect a better rate of convergence. For example, Newton's method, which uses the Hessian of the function in calculation of the search direction, has a *quadratic* rate of convergence (meaning it converges very rapidly when the design point is within certain radius of the minimum point). For any positive definite quadratic function, the method converges in just one iteration with a step size of one.

9.4.1 Classical Newton's Method

The *basic idea* of the Newton's method is to use a second-order Taylor's expansion of the function about the current design point. This gives a quadratic expression for the change in design $\Delta\mathbf{x}$. The necessary condition for minimization of this function then gives an explicit calculation for design change. In the following, we shall omit the argument $\mathbf{x}^{(k)}$ from all functions, because the derivation applies to any design iteration. Using second-order Taylor's expansion for the function $f(\mathbf{x})$, we obtain

$$f(\mathbf{x} + \Delta\mathbf{x}) = f(\mathbf{x}) + \mathbf{c}^T \Delta\mathbf{x} + 0.5 \Delta\mathbf{x}^T \mathbf{H} \Delta\mathbf{x} \quad (9.7)$$

where $\Delta\mathbf{x}$ is a small change in design and \mathbf{H} is the Hessian of f at the point \mathbf{x} (sometimes denoted as $\nabla^2 f$). Equation (9.7) is a quadratic function in terms of $\Delta\mathbf{x}$. The theory of convex programming problems in Chapter 4 guarantees that if \mathbf{H} is positive semidefinite, then there is a $\Delta\mathbf{x}$ that gives a global minimum for the function of Eq. (9.7). In addition, if \mathbf{H} is positive definite, then the minimum for Eq. (9.7) is unique. Writing optimality conditions $[\partial f / \partial(\Delta\mathbf{x}) = \mathbf{0}]$ for the function of Eq. (9.7),

$$\mathbf{c} + \mathbf{H}\Delta\mathbf{x} = \mathbf{0} \quad (9.8)$$

Assuming \mathbf{H} to be nonsingular, we get an expression for $\Delta\mathbf{x}$ as

$$\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{c} \quad (9.9)$$

Using this value for $\Delta\mathbf{x}$, the design is updated as

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \Delta\mathbf{x} \quad (9.10)$$

Since Eq. (9.7) is just an approximation for f at the point $\mathbf{x}^{(0)}$, $\mathbf{x}^{(1)}$ will probably not be the precise minimum point of $f(\mathbf{x})$. Therefore, the process will have to be repeated to obtain improved estimates until the minimum is reached. Each iteration of Newton's method requires computation of the Hessian of the cost function. Since it is a symmetric matrix, it needs computation of $n(n+1)/2$ second-order derivatives of $f(\mathbf{x})$ (recall that n is the number of design variables). This can require considerable computational effort.

9.4.2 Modified Newton's Method

Note that the classical Newton's method does not have a step size associated with the calculation of design change $\Delta\mathbf{x}$ in Eq. (9.9); i.e., step size is taken as one (step of length one is called an *ideal step size* or *Newton's step*). Therefore, there is no way to ensure that the cost function will be reduced at each iteration; i.e., to ensure that $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$. Thus, the method is not guaranteed to converge to a local minimum point even with the use of second-order information that requires large calculations. This situation can be corrected if we incorporate the use of a step size in the calculation of the design change $\Delta\mathbf{x}$. In other words, we treat the solution of Eq. (9.9) as the search direction and use any of the one-dimensional search methods to calculate the step size in the search direction. This is called the *modified Newton's method* and is stated as follows.

- Step 1.* Make an engineering estimate for a starting design $\mathbf{x}^{(0)}$. Set iteration counter $k = 0$. Select a tolerance ε for the stopping criterion.
- Step 2.* Calculate $c_i^{(k)} = \partial f(\mathbf{x}^{(k)})/\partial x_i$ for $i = 1$ to n . If $\|\mathbf{c}^{(k)}\| < \varepsilon$, stop the iterative process. Otherwise, continue.
- Step 3.* Calculate the Hessian matrix $\mathbf{H}^{(k)}$ at the current point $\mathbf{x}^{(k)}$.
- Step 4.* Calculate the search by solving Eq. (9.9) as

$$\mathbf{d}^{(k)} = -[\mathbf{H}^{(k)}]^{-1} \mathbf{c}^{(k)} \quad (9.11)$$

Note that the calculation of $\mathbf{d}^{(k)}$ in the above equation is symbolic. For computational efficiency, the linear equation $\mathbf{H}^{(k)}\mathbf{d}^{(k)} = -\mathbf{c}^{(k)}$ is solved directly instead of evaluating the inverse of the Hessian matrix.

- Step 5.* Update the design as $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$, where α_k is calculated to minimize $f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$. Any one-dimensional search procedure may be used to calculate α .
- Step 6.* Set $k = k + 1$ and go to Step 2.

It is important to note here that unless \mathbf{H} is positive definite, the direction $\mathbf{d}^{(k)}$ determined from Eq. (9.11) may not be that of descent for the cost function. To see this, we substitute $\mathbf{d}^{(k)}$ from Eq. (9.11) into the descent condition of Eq. (8.8) to obtain

$$-\mathbf{c}^{(k)T} \mathbf{H}^{-1} \mathbf{c}^{(k)} < 0 \quad (9.12)$$

The foregoing condition will always be satisfied if \mathbf{H} is positive definite. If \mathbf{H} is negative definite or negative semidefinite, the condition is always violated. With \mathbf{H} as indefinite or positive semidefinite, the condition may or may not be satisfied, so we must check for it. If the direction obtained in Step 4 is not that of descent for the cost function, then we should stop there because a positive step size cannot be determined. Based on the foregoing discussion, it is suggested that the descent condition of Eq. (8.8) should be checked for Newton's search direction at each iteration before calculating the step size. Examples 9.6 and 9.7 demonstrate use of the modified Newton's method.

EXAMPLE 9.6 Use of Modified Newton's Method

$$\text{Minimize } f(\mathbf{x}) = 3x_1^2 + 2x_1x_2 + 2x_2^2 + 7 \quad (\text{a})$$

using the modified Newton's algorithm starting from the point (5, 10). Use $\varepsilon = 0.0001$ as the stopping criterion.

Solution. We will follow the steps of the modified Newton's method.

1. $\mathbf{x}^{(0)}$ is given as (5, 10).
2. The gradient vector $\mathbf{c}^{(0)}$ at the point (5, 10) is given as

$$\mathbf{c}^{(0)} = (6x_1 + 2x_2, 2x_1 + 4x_2) = (50, 50) \quad (\text{b})$$

$$\|\mathbf{c}^{(0)}\| = \sqrt{50^2 + 50^2} = 50\sqrt{2} > \varepsilon \quad (\text{c})$$

Therefore, the convergence criterion is not satisfied.

3. The Hessian matrix at the point (5, 10) is given as

$$\mathbf{H}^{(0)} = \begin{bmatrix} 6 & 2 \\ 2 & 4 \end{bmatrix} \quad (\text{d})$$

Note that the Hessian does not depend on design variables and is positive definite (since its eigenvalues are 7.24 and 2.76). Therefore Newton's direction satisfies the descent condition at each iteration.

4. The direction of design change is

$$\mathbf{d}^{(0)} = -\mathbf{H}^{-1}\mathbf{c}^{(0)} = \frac{-1}{20} \begin{bmatrix} 4 & -2 \\ -2 & 6 \end{bmatrix} \begin{bmatrix} 50 \\ 50 \end{bmatrix} = \begin{bmatrix} -5 \\ -10 \end{bmatrix} \quad (\text{e})$$

5. Step size α is calculated to minimize $f(\mathbf{x}^{(0)} + \alpha\mathbf{d}^{(0)})$:

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha\mathbf{d}^{(0)} = \begin{bmatrix} 5 \\ 10 \end{bmatrix} + \alpha \begin{bmatrix} -5 \\ -10 \end{bmatrix} = \begin{bmatrix} 5 - 5\alpha \\ 10 - 10\alpha \end{bmatrix} \quad (\text{f})$$

$$\frac{df}{d\alpha} = 0; \quad \text{or} \quad \nabla f(\mathbf{x}^{(1)}) \cdot \mathbf{d}^{(0)} = 0 \quad (\text{g})$$

Using the Step 2 calculations, $\nabla f(\mathbf{x}^{(1)})$ and the dot product $\nabla f(\mathbf{x}^{(1)}) \cdot \mathbf{d}^{(0)}$ are calculated as

$$\nabla f(\mathbf{x}^{(1)}) = \begin{bmatrix} 6(5 - 5\alpha) + 2(10 - 10\alpha) \\ 2(5 - 5\alpha) + 4(10 - 10\alpha) \end{bmatrix} = \begin{bmatrix} 50 - 50\alpha \\ 50 - 50\alpha \end{bmatrix} \quad (\text{h})$$

$$\nabla f(\mathbf{x}^{(1)}) \cdot \mathbf{d}^{(0)} = (50 - 50\alpha, 50 - 50\alpha) \begin{bmatrix} -5 \\ -10 \end{bmatrix} = 0 \quad (\text{i})$$

$$\text{Or, } -5(50 - 50\alpha) - 10(50 - 50\alpha) = 0 \quad (\text{j})$$

Solving the preceding equation, we get $\alpha = 1$. Note that the golden section search also gives $\alpha = 1$. Therefore,

$$\mathbf{x}^{(1)} = \begin{bmatrix} 5 - 5\alpha \\ 10 - 10\alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (k)$$

The gradient of the cost function at $\mathbf{x}^{(1)}$ is calculated as

$$\mathbf{c}^{(1)} = \begin{bmatrix} 50 - 50\alpha \\ 50 - 50\alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (l)$$

Since $\|\mathbf{c}^{(k)}\| < \varepsilon$, the Newton's method has given the solution in just one iteration. This is because the function is a positive definite quadratic form (the Hessian of f is positive definite everywhere). Note that the condition number of the Hessian is not 1; therefore the steepest descent method will not converge in one iteration, as was the case in Examples 9.4 and 9.5.

A computer program based on the modified Newton's method is given in Appendix D, which needs three user-supplied subroutines FUNCT, GRAD, and HASN. These subroutines evaluate cost function, the gradient, and the Hessian matrix of the cost function, respectively. The program is used to solve the problem of Example 9.7.

EXAMPLE 9.7 Use of Modified Newton's Method

$$\text{Minimize } f(\mathbf{x}) = 10x_1^4 - 20x_1^2x_2 + 10x_2^2 + x_1^2 - 2x_1 + 5 \quad (a)$$

using the computer program for the modified Newton's method given in Appendix D from the point $(-1, 3)$. Golden section search may be used for step size determination with $\delta = 0.05$ and line search accuracy equal to 0.0001. For the stopping criterion, use $\varepsilon = 0.005$.

Solution. Note that $f(\mathbf{x})$ is not a quadratic function in terms of the design variables. Thus, we cannot expect the Newton's method to converge in one iteration. The gradient of $f(\mathbf{x})$ is given as

$$\mathbf{c} = \nabla f(\mathbf{x}) = (40x_1^3 - 40x_1x_2 + 2x_1 - 2, -20x_1^2 + 20x_2) \quad (b)$$

and the Hessian matrix of $f(\mathbf{x})$ is

$$\mathbf{H} = \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 120x_1^2 - 40x_2 + 2 & -40x_1 \\ -40x_1 & 20 \end{bmatrix} \quad (c)$$

Results with the modified Newton's method for the problem are given in Table 9-2. The optimum point is $(1, 1)$ and the optimum value of $f(\mathbf{x})$ is 4.0. Newton's method has converged to the optimum solution in eight iterations. Figure 9-7 shows the contours for the function and the progress of the method from the starting design $(-1, 3)$. It is noted that the step size was approximately equal to one in the last phase of the iterative process. This is because the function resembles a quadratic function sufficiently close to the optimum point and step size is equal to unity for a quadratic function.

TABLE 9-2 Optimum Solution for Example 9.7 with Modified Newton's Method: $f(\mathbf{x}) = 10x_1^4 - 20x_1^2x_2 + 10x_2^2 + x_1^2 - 2x_1 + 5$

Starting point:	-1, 3
Optimum design variables:	9.99880E-01, 9.99681E-01
Optimum cost function value:	4.0
Norm of gradient at optimum:	3.26883E-03
Number of iterations:	8
Number of function evaluations:	198

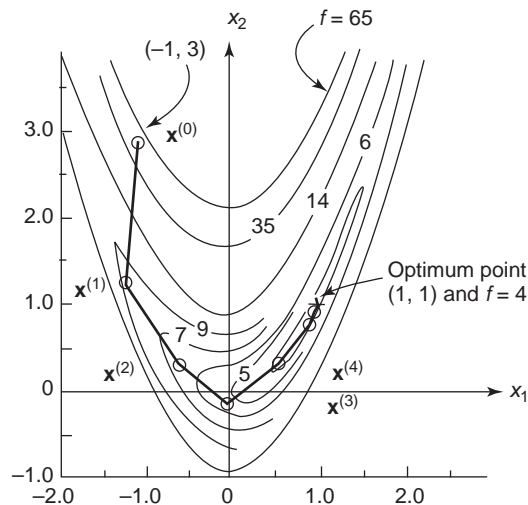


FIGURE 9-7 Iteration history for Example 9.7 with Newton's method.

The *drawbacks* of the modified Newton's method for general applications are:

1. It requires calculations of second-order derivatives at each iteration, which is usually quite time consuming. In some applications it may not even be possible to calculate such derivatives. Also, a linear system of equations in Eq. (9.11) needs to be solved. Therefore, each iteration of the method requires substantially more calculations compared with the steepest descent or conjugate gradient method.
2. The Hessian of the cost function may be singular at some iterations. Thus, Eq. (9.11) cannot be used to compute the search direction. Also, unless the Hessian is positive definite, the search direction cannot be guaranteed to be that of descent for the cost function, as discussed earlier.
3. The method is not convergent unless the Hessian remains positive definite and a step size is calculated along the search direction to update design. However, the method has a quadratic rate of convergence when it converges. For a strictly convex quadratic function, the method converges in just one iteration from any starting design.

A comparison of steepest descent, conjugate gradient, and modified Newton methods is presented in Example 9.8.

EXAMPLE 9.8 Comparison of Steepest Descent, Conjugate Gradient, and Modified Newton Methods

Minimize $f(\mathbf{x}) = 50(x_2 - x_1^2)^2 + (2 - x_1)^2$ starting from the point $(5, -5)$. Use the steepest descent, Newton, and conjugate gradient methods, and compare their performance.

Solution. The minimum point for the function is known as $(2, 4)$ with $f(2, 4) = 0$. We use exact gradient expressions and $\varepsilon = 0.005$ to solve the problem using the steepest descent and Newton's method programs given in Appendix D and the conjugate gradient method available in IDESIGN. Table 9-3 summarizes final results with the three methods. For the steepest descent method, $\delta_0 = 0.05$ and a line search termination criterion of 0.00001 are used. For the Newton's method, they are 0.05 and 0.0001 respectively. Golden section search is used with both methods. It can be observed again that for the present example the steepest descent method is the most inefficient and the conjugate gradient is most efficient. Therefore, the conjugate gradient method is recommended for general applications.

TABLE 9-3 Evaluation of Three Methods for Example 9.8: $f(\mathbf{x}) = 50(x_2 - x_1^2)^2 + (2 - x_1)^2$

	<i>Steepest descent</i>	<i>Conjugate gradient</i>	<i>Modified Newton</i>
x_1	1.9941	2.0000	2.0000
x_2	3.9765	3.9998	3.9999
f	3.4564E-05	1.0239E-08	2.5054E-10
$\ \mathbf{c}\ $	3.3236E-03	1.2860E-04	9.0357E-04
No. of function evaluations	138,236	65	349
No. of iterations	9670	22	13

9.4.3 Marquardt Modification

As noted before the modified Newton's method has several drawbacks that can cause numerical difficulties. For example, if the Hessian \mathbf{H} of the cost function is not positive definite, the direction found from Eq. (9.11) may not be that of descent for the cost function. In that case, a step cannot be executed along the direction. Marquardt (1963) suggested a modification to the direction finding process that has the desirable features of the steepest descent and Newton's methods. It turns out that far away from the solution point, the method behaves like the steepest descent method, which is quite good there. Near the solution point, it behaves like the Newton's method, which is very effective there. In the modified procedure, the Hessian is modified as $(\mathbf{H} + \lambda\mathbf{I})$, where λ is a positive constant. λ is initially selected as a large number that is reduced as iterations progress. The search direction is computed from Eq. (9.11) as

$$\mathbf{d}^{(k)} = -[\mathbf{H}^{(k)} + \lambda_k \mathbf{I}]^{-1} \mathbf{c}^{(k)} \quad (9.13)$$

Note that when λ is large, the effect of \mathbf{H} essentially gets neglected and $\mathbf{d}^{(k)}$ is essentially $-(1/\lambda)\mathbf{c}^{(k)}$, which is the steepest descent direction with $1/\lambda$ as the step size. As the algorithm proceeds, λ is reduced (i.e., step size is increased). When λ becomes sufficiently small, then the effect of $\lambda\mathbf{I}$ is essentially neglected and the Newton direction is obtained from Eq. (9.13). If the direction $\mathbf{d}^{(k)}$ of Eq. (9.13) does not reduce the cost function, then λ is increased (step

size is reduced) and the search direction is recomputed. *Marquardt's algorithm* is summarized in the following steps.

- Step 1.* Make an engineering estimate for starting design $\mathbf{x}^{(0)}$. Set iteration counter $k = 0$.
Select a tolerance ε as the stopping criterion, and λ_0 as a large constant (say 1000).
- Step 2.* Calculate $c_i^{(k)} = \partial f(\mathbf{x}^{(k)})/\partial x_i$ for $i = 1$ to n . If $\|\mathbf{c}^{(k)}\| < \varepsilon$, stop. Otherwise, continue.
- Step 3.* Calculate the Hessian matrix $\mathbf{H}(\mathbf{x}^{(k)})$.
- Step 4.* Calculate the search direction by solving Eq. (9.13).
- Step 5.* If $f(\mathbf{x}^{(k)} + \mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)})$, then continue. Otherwise, increase λ_k (to say $2\lambda_k$), and go to Step 4.
- Step 6.* Reduce λ_k , say, $\lambda_{k+1} = 0.5\lambda_k$. Set $k = k + 1$ and go to Step 2.

9.5 Search Direction Determination: Quasi-Newton Methods

In Section 8.3 the steepest descent method was described. Some of the drawbacks of that method were pointed out. It was noted that the method has a poor rate of convergence because only first-order information is used. This flaw was corrected with Newton's method where second-order derivatives were used. Newton's method has very good convergence properties. However, the method can be inefficient because it requires calculation of $n(n + 1)/2$ second-order derivatives to generate the Hessian matrix (recall that n is the number of design variables). For most engineering design problems, calculation of second-order derivatives may be tedious or even impossible. Also, Newton's method runs into difficulties if the Hessian of the function is singular at any iteration. The methods presented in this section overcome these drawbacks 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. Therefore the methods have desirable features of both the steepest descent and the Newton's methods. They are called *quasi-Newton methods*.

The quasi-Newton methods were initially developed for positive definite quadratic functions. For such functions they converge to the exact optimum in at most n iterations. However, this ideal behavior does not carry over to general cost functions, and the methods usually need to be restarted at every $(n + 1)$ th iteration.

There are several ways to approximate the Hessian or its inverse. The basic idea is to update the current approximation of the Hessian using two pieces of information: change in design and the gradient vectors between two successive iterations. While updating, the properties of symmetry and positive definiteness are preserved. Positive definiteness is essential because without that the search direction may not be a descent direction for the cost function. The derivation of the updating procedures is based on the so-called quasi-Newton condition (Gill *et al.*, 1981). This condition is derived by requiring the curvature of the cost function in the search direction $\mathbf{d}^{(k)}$ to be the same at two consecutive points $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(k+1)}$. The enforcement of this condition gives the updating formulas for the Hessian of the cost function or its inverse. For a strictly convex quadratic function, the updating procedure converges to the exact Hessian in n iterations. We shall describe two of the most popular methods in the class of quasi-Newton methods.

9.5.1 Inverse Hessian Updating: DFP Method

This method, initially proposed by Davidon (1959), was modified by Fletcher and Powell (1963) and that method is presented here. This is one of the most powerful methods for the minimization of a general function $f(\mathbf{x})$. The method builds an *approximate inverse of the Hessian* of $f(\mathbf{x})$ using only the first derivatives. It is often called the DFP (Davidon-Fletcher-Powell) method:

Step 1. Estimate an initial design $\mathbf{x}^{(0)}$. Choose a symmetric positive definite $n \times n$ matrix $\mathbf{A}^{(0)}$ as an estimate for the inverse of the Hessian of the cost function. In the absence of more information, $\mathbf{A}^{(0)} = \mathbf{I}$ may be chosen. Also, specify a convergence parameter ε . Set $k = 0$. Compute the gradient vector as $\mathbf{c}^{(0)} = \nabla f(\mathbf{x}^{(0)})$.

Step 2. Calculate the norm of the gradient vector as $\|\mathbf{c}^{(k)}\|$. If $\|\mathbf{c}^{(k)}\| < \varepsilon$, then stop the iterative process. Otherwise continue.

Step 3. Calculate the search direction as $\mathbf{d}^{(k)} = -\mathbf{A}^{(k)}\mathbf{c}^{(k)}$ (a)

Step 4. Compute optimum step size $\alpha_k = \alpha$ to minimize $f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$.

Step 5. Update the design as $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)}$ (b)

Step 6. Update the matrix $\mathbf{A}^{(k)}$ —approximation for the inverse of the Hessian of the cost function—as

$$\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} + \mathbf{B}^{(k)} + \mathbf{C}^{(k)} \quad (c)$$

where the correction matrices $\mathbf{B}^{(k)}$ and $\mathbf{C}^{(k)}$ are calculated using the quasi-Newton condition mentioned earlier, as

$$\mathbf{B}^{(k)} = \frac{\mathbf{s}^{(k)}\mathbf{s}^{(k)T}}{(\mathbf{s}^{(k)} \cdot \mathbf{y}^{(k)})}; \quad \mathbf{C}^{(k)} = \frac{-\mathbf{z}^{(k)}\mathbf{z}^{(k)T}}{(\mathbf{y}^{(k)} \cdot \mathbf{z}^{(k)})} \quad (d)$$

$$\mathbf{s}^{(k)} = \alpha_k\mathbf{d}^{(k)} \text{ (change in design);} \quad \mathbf{y}^{(k)} = \mathbf{c}^{(k+1)} - \mathbf{c}^{(k)} \text{ (change in gradient)} \quad (e)$$

$$\mathbf{c}^{(k+1)} = \nabla f(\mathbf{x}^{(k+1)}); \quad \mathbf{z}^{(k)} = \mathbf{A}^{(k)}\mathbf{y}^{(k)} \quad (f)$$

Step 7. Set $k = k + 1$ and go to Step 2.

Note that the first iteration of the method is the same as that for the steepest descent method. Fletcher and Powell (1963) prove that this algorithm has the following properties:

1. The matrix $\mathbf{A}^{(k)}$ is positive definite for all k . This implies that the method will always converge to a local minimum point, since

$$\left. \frac{d}{d\alpha} f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}) \right|_{\alpha=0} = -\mathbf{c}^{(k)T}\mathbf{A}^{(k)}\mathbf{c}^{(k)} < 0 \quad (g)$$

as long as $\mathbf{c}^{(k)} \neq \mathbf{0}$. This means that $f(\mathbf{x}^{(k)})$ may be decreased by choosing $\alpha > 0$ if $\mathbf{c}^{(k)} \neq \mathbf{0}$ (i.e., $\mathbf{d}^{(k)}$ is a direction of descent).

2. When this method is applied to a positive definite quadratic form, $\mathbf{A}^{(k)}$ converges to the inverse of the Hessian of the quadratic form.

Example 9.9 illustrates calculations for two iterations of the DFP method.

EXAMPLE 9.9 Application of DFP Method

Execute two iterations of the DFP method for the problem: minimize $f(\mathbf{x}) = 5x_1^2 + 2x_1x_2 + x_2^2 + 7$ starting from the point (1, 2).

Solution. We shall follow steps of the algorithm.

Iteration 1 ($k = 0$).

1. $\mathbf{x}^{(0)} = (1, 2)$; $\mathbf{A}^{(0)} = \mathbf{I}$, $k = 0$, $\varepsilon = 0.001$
 $\mathbf{c}^{(0)} = (10x_1 + 2x_2, 2x_1 + 2x_2) = (14, 6)$
2. $\|\mathbf{c}^{(0)}\| = \sqrt{14^2 + 6^2} = 15.232 > \varepsilon$, so continue
3. $\mathbf{d}^{(0)} = -\mathbf{c}^{(0)} = (-14, -6)$
4. $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha \mathbf{d}^{(0)} = (1 - 14\alpha, 2 - 6\alpha)$

$$f(\mathbf{x}^{(1)}) = f(\alpha) = 5(1 - 14\alpha)^2 + 2(1 - 14\alpha)(2 - 6\alpha) + (2 - 6\alpha)^2 + 7 \quad (\text{a})$$

$$\begin{aligned} \frac{df}{d\alpha} &= 5(2)(-14)(1 - 14\alpha) + 2(-14)(2 - 6\alpha) + 2(-6)(1 - 14\alpha) \\ &\quad + 2(-6)(2 - 6\alpha) = 0 \end{aligned} \quad (\text{b})$$

$$\alpha_0 = 0.099$$

$$\frac{d^2f}{d\alpha^2} = 2348 > 0. \text{ Therefore, step size of } \alpha = 0.099 \text{ is acceptable.}$$

5. $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)} = (-0.386, 1.407)$
6. $\mathbf{s}^{(0)} = \alpha_0 \mathbf{d}^{(0)} = (-1.386, -0.593)$; $\mathbf{c}^{(1)} = (-1.046, 2.042)$ (c)

$$\mathbf{y}^{(0)} = \mathbf{c}^{(1)} - \mathbf{c}^{(0)} = (-15.046, -3.958); \quad \mathbf{z}^{(0)} = \mathbf{y}^{(0)} = (-15.046, -3.958) \quad (\text{d})$$

$$\mathbf{s}^{(0)} \cdot \mathbf{y}^{(0)} = 23.20; \quad \mathbf{y}^{(0)} \cdot \mathbf{z}^{(0)} = 242.05 \quad (\text{e})$$

$$\mathbf{s}^{(0)} \mathbf{s}^{(0)T} = \begin{bmatrix} 1.921 & 0.822 \\ 0.822 & 0.352 \end{bmatrix}; \quad \mathbf{B}^{(0)} = \frac{\mathbf{s}^{(0)} \mathbf{s}^{(0)T}}{\mathbf{s}^{(0)} \cdot \mathbf{y}^{(0)}} = \begin{bmatrix} 0.0828 & 0.0354 \\ 0.0354 & 0.0152 \end{bmatrix} \quad (\text{f})$$

$$\mathbf{z}^{(0)} \mathbf{z}^{(0)T} = \begin{bmatrix} 226.40 & 59.55 \\ 59.55 & 15.67 \end{bmatrix}; \quad \mathbf{C}^{(0)} = -\frac{\mathbf{z}^{(0)} \mathbf{z}^{(0)T}}{\mathbf{y}^{(0)} \cdot \mathbf{z}^{(0)}} = \begin{bmatrix} -0.935 & -0.246 \\ -0.246 & -0.065 \end{bmatrix} \quad (\text{g})$$

$$\mathbf{A}^{(1)} = \mathbf{A}^{(0)} + \mathbf{B}^{(0)} + \mathbf{C}^{(0)} = \begin{bmatrix} 0.148 & -0.211 \\ -0.211 & 0.950 \end{bmatrix} \quad (\text{h})$$

Iteration 2 ($k = 1$).

2. $\|\mathbf{c}^{(1)}\| = 2.29 > \varepsilon$, so continue
3. $\mathbf{d}^{(1)} = -\mathbf{A}^{(1)} \mathbf{c}^{(1)} = (0.586, -1.719)$
4. Step size determination: minimize $f(\mathbf{x}^{(1)} + \alpha \mathbf{d}^{(1)})$; $\alpha_1 = 0.776$
5. $\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha_1 \mathbf{d}^{(1)} = (-0.386, 1.407) + (0.455, -1.334) = (0.069, 0.073)$
6. $\mathbf{s}^{(1)} = \alpha_1 \mathbf{d}^{(1)} = (0.455, -1.334)$

$$\mathbf{c}^{(2)} = (0.836, 0.284); \quad \mathbf{y}^{(1)} = \mathbf{c}^{(2)} - \mathbf{c}^{(1)} = (1.882, -1.758) \quad (\text{i})$$

$$\mathbf{z}^{(1)} = \mathbf{A}^{(1)} \mathbf{y}^{(1)} = (0.649, -2.067); \quad \mathbf{s}^{(1)} \cdot \mathbf{y}^{(1)} = 3.201; \quad \mathbf{y}^{(1)} \cdot \mathbf{z}^{(1)} = 4.855 \quad (\text{j})$$

$$\mathbf{s}^{(1)} \mathbf{s}^{(1)T} = \begin{bmatrix} 0.207 & -0.607 \\ -0.607 & 1.780 \end{bmatrix}; \quad \mathbf{B}^{(1)} = \frac{\mathbf{s}^{(1)} \mathbf{s}^{(1)T}}{\mathbf{s}^{(1)} \cdot \mathbf{y}^{(1)}} = \begin{bmatrix} 0.0647 & -0.19 \\ -0.19 & 0.556 \end{bmatrix} \quad (\text{k})$$

$$\mathbf{z}^{(1)} \mathbf{z}^{(1)T} = \begin{bmatrix} 0.421 & -1.341 \\ -1.341 & 4.272 \end{bmatrix}; \quad \mathbf{C}^{(1)} = -\frac{\mathbf{z}^{(1)} \mathbf{z}^{(1)T}}{\mathbf{y}^{(1)} \cdot \mathbf{z}^{(1)}} = \begin{bmatrix} -0.0867 & 0.276 \\ 0.276 & -0.880 \end{bmatrix} \quad (\text{l})$$

$$\mathbf{A}^{(2)} = \mathbf{A}^{(1)} + \mathbf{B}^{(1)} + \mathbf{C}^{(1)} = \begin{bmatrix} 0.126 & -0.125 \\ -0.125 & 0.626 \end{bmatrix} \quad (\text{m})$$

It can be verified that the matrix $\mathbf{A}^{(2)}$ is quite close to the inverse of the Hessian of the cost function. One more iteration of the DFP method will yield the optimum solution of (0, 0).

9.5.2 Direct Hessian Updating: BFGS Method

It is possible to update the Hessian rather than its inverse at every iteration. Several such updating methods are available; however, we shall present a popular method that has proven to be most effective in applications. Detailed derivation of the method is given in Gill and coworkers (1981). It is known as the *Broyden-Fletcher-Goldfarb-Shanno (BFGS) method*, which is summarized in the following algorithm:

- Step 1.* Estimate an initial design $\mathbf{x}^{(0)}$. Choose a symmetric positive definite $n \times n$ matrix $\mathbf{H}^{(0)}$ as an estimate for the Hessian of the cost function. In the absence of more information, let $\mathbf{H}^{(0)} = \mathbf{I}$. Choose a convergence parameter ε . Set $k = 0$, and compute the gradient vector as $\mathbf{c}^{(0)} = \nabla f(\mathbf{x}^{(0)})$.
- Step 2.* Calculate the norm of the gradient vector as $\|\mathbf{c}^{(k)}\|$. If $\|\mathbf{c}^{(k)}\| < \varepsilon$ then stop the iterative process; otherwise continue.
- Step 3.* Solve the linear system of equations $\mathbf{H}^{(k)}\mathbf{d}^{(k)} = -\mathbf{c}^{(k)}$ to obtain the search direction.
- Step 4.* Compute optimum step size $\alpha_k = \alpha$ to minimize $f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})$.
- Step 5.* Update the design as $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)}$
- Step 6.* Update the Hessian approximation for the cost function as

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \mathbf{D}^{(k)} + \mathbf{E}^{(k)} \quad (\text{a})$$

where the correction matrices $\mathbf{D}^{(k)}$ and $\mathbf{E}^{(k)}$ are given as

$$\mathbf{D}^{(k)} = \frac{\mathbf{y}^{(k)}\mathbf{y}^{(k)T}}{(\mathbf{y}^{(k)} \cdot \mathbf{s}^{(k)})}; \quad \mathbf{E}^{(k)} = \frac{\mathbf{c}^{(k)}\mathbf{c}^{(k)T}}{(\mathbf{c}^{(k)} \cdot \mathbf{d}^{(k)})} \quad (\text{b})$$

$$\begin{aligned} \mathbf{s}^{(k)} &= \alpha_k\mathbf{d}^{(k)} \text{ (change in design);} & \mathbf{y}^{(k)} &= \mathbf{c}^{(k+1)} - \mathbf{c}^{(k)} \text{ (change in gradient);} \\ \mathbf{c}^{(k+1)} &= \nabla f(\mathbf{x}^{(k+1)}) \end{aligned} \quad (\text{c})$$

- Step 7.* Set $k = k + 1$ and go to Step 2.

Note again that the first iteration of the method is the same as that for the steepest descent method when $\mathbf{H}^{(0)} = \mathbf{I}$. It can be shown that the BFGS update formula keeps the Hessian approximation positive definite if an accurate line search is used. This is important to know because the search direction is guaranteed to be that of descent for the cost function if $\mathbf{H}^{(k)}$ is positive definite. In numerical calculations, difficulties can arise because the Hessian can become singular or indefinite as a result of inaccurate line search, and round-off and truncation errors. Therefore, some safeguards against the numerical difficulties must be incorporated into computer programs for stable and convergent calculations. Another numerical procedure that is extremely useful is to update decomposed factors (Cholesky factors) of the Hessian rather than the Hessian itself. With that procedure, the matrix can numerically be guaranteed to be positive definite, and the linear equation $\mathbf{H}^{(k)}\mathbf{d}^{(k)} = -\mathbf{c}^{(k)}$ can be solved more efficiently.

Example 9.10 illustrates calculations for two iterations of the BFGS method.

EXAMPLE 9.10 Application of the BFGS Method

Execute two iterations of the BFGS method for the problem:

$$\text{minimize } f(\mathbf{x}) = 5x_1^2 + 2x_1x_2 + x_2^2 + 7 \text{ starting from the point } (1, 2).$$

Solution. We shall follow steps of the algorithm. Note that the first iteration gives steepest descent step for the cost function.

Iteration 1 (k = 0).

1. $\mathbf{x}^{(0)}(1, 2)$, $\mathbf{H}^{(0)} = \mathbf{I}$, $\varepsilon = 0.001$, $k = 0$
 $\mathbf{c}^{(0)} = (10x_1 + 2x_2, 2x_1 + 2x_2) = (14, 6)$
2. $\|\mathbf{c}^{(0)}\| = \sqrt{14^2 + 6^2} = 15.232 > \varepsilon$, so continue
3. $\mathbf{d}^{(0)} = -\mathbf{c}^{(0)} = (-14, -6)$; since $\mathbf{H}^{(0)} = \mathbf{I}$
4. Step size determination (same as Example 9.9): $\alpha_0 = 0.099$
5. $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)} = (-0.386, 1.407)$
6. $\mathbf{s}^{(0)} = \alpha_0 \mathbf{d}^{(0)} = (-1.386, 0.593)$; $\mathbf{c}^{(1)} = (-1.046, 2.042)$

$$\mathbf{y}^{(0)} = \mathbf{c}^{(1)} - \mathbf{c}^{(0)} = (-15.046, -3.958); \quad \mathbf{y}^{(0)} \cdot \mathbf{s}^{(0)} = 23.20; \quad \mathbf{c}^{(0)} \cdot \mathbf{d}^{(0)} = -232.0 \quad (\text{a})$$

$$\mathbf{y}^{(0)} \mathbf{y}^{(0)T} = \begin{bmatrix} 226.40 & 59.55 \\ 59.55 & 15.67 \end{bmatrix}; \quad \mathbf{D}^{(0)} = \frac{\mathbf{y}^{(0)} \mathbf{y}^{(0)T}}{\mathbf{y}^{(0)} \cdot \mathbf{s}^{(0)}} = \begin{bmatrix} 9.760 & 2.567 \\ 2.567 & 0.675 \end{bmatrix} \quad (\text{b})$$

$$\mathbf{c}^{(0)} \mathbf{c}^{(0)T} = \begin{bmatrix} 196 & 84 \\ 84 & 36 \end{bmatrix}; \quad \mathbf{E}^{(0)} = \frac{\mathbf{c}^{(0)} \mathbf{c}^{(0)T}}{\mathbf{c}^{(0)} \cdot \mathbf{d}^{(0)}} = \begin{bmatrix} -0.845 & -0.362 \\ -0.362 & -0.155 \end{bmatrix} \quad (\text{c})$$

$$\mathbf{H}^{(1)} = \mathbf{H}^{(0)} + \mathbf{D}^{(0)} + \mathbf{E}^{(0)} = \begin{bmatrix} 9.915 & 2.205 \\ 2.205 & 0.520 \end{bmatrix} \quad (\text{d})$$

Iteration 2 (k = 1).

2. $\|\mathbf{c}^{(1)}\| = 2.29 > \varepsilon$, so continue
3. $\mathbf{H}^{(1)} \mathbf{d}^{(1)} = -\mathbf{c}^{(1)}$; or, $\mathbf{d}^{(1)} = (17.20, -76.77)$
4. Step size determination: $\alpha_1 = 0.018455$
5. $\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha_1 \mathbf{d}^{(1)} = (-0.0686, -0.0098)$
6. $\mathbf{s}^{(1)} = \alpha_1 \mathbf{d}^{(1)} = (0.317, -1.417)$; $\mathbf{c}^{(2)} = (-0.706, -0.157)$ (e)

$$\mathbf{y}^{(1)} = \mathbf{c}^{(2)} - \mathbf{c}^{(1)} = (0.317, -2.199); \quad \mathbf{y}^{(1)} \cdot \mathbf{s}^{(1)} = 3.224; \quad \mathbf{c}^{(1)} \cdot \mathbf{d}^{(1)} = -174.76 \quad (\text{f})$$

$$\mathbf{y}^{(1)} \mathbf{y}^{(1)T} = \begin{bmatrix} 0.1156 & -0.748 \\ -0.748 & 4.836 \end{bmatrix}; \quad \mathbf{D}^{(1)} = \frac{\mathbf{y}^{(1)} \mathbf{y}^{(1)T}}{\mathbf{y}^{(1)} \cdot \mathbf{s}^{(1)}} = \begin{bmatrix} 0.036 & -0.232 \\ -0.232 & 1.500 \end{bmatrix} \quad (\text{g})$$

$$\mathbf{c}^{(1)} \mathbf{c}^{(1)T} = \begin{bmatrix} 1.094 & -2.136 \\ -2.136 & 4.170 \end{bmatrix}; \quad \mathbf{E}^{(1)} = \frac{\mathbf{c}^{(1)} \mathbf{c}^{(1)T}}{\mathbf{c}^{(1)} \cdot \mathbf{d}^{(1)}} = \begin{bmatrix} -0.0063 & 0.0122 \\ 0.0122 & -0.0239 \end{bmatrix} \quad (\text{h})$$

$$\mathbf{H}^{(2)} = \mathbf{H}^{(1)} + \mathbf{D}^{(1)} + \mathbf{E}^{(1)} = \begin{bmatrix} 9.945 & 1.985 \\ 1.985 & 1.996 \end{bmatrix} \quad (\text{i})$$

It can be verified that $\mathbf{H}^{(2)}$ is quite close to the Hessian of the given cost function. One more iteration of the BFGS method will yield the optimum solution of (0, 0).

9.6 Engineering Applications of Unconstrained Methods

There are several engineering applications where unconstrained optimization methods can be used. For example, linear as well as nonlinear simultaneous equations can be solved with unconstrained optimization methods. Such equations arise while calculating the response of structural and mechanical systems. The procedures have been incorporated into some commercial software packages as well.

9.6.1 Minimization of Total Potential Energy

The equilibrium states of structural and mechanical systems are characterized by the stationary points of the total potential energy of the system. This is known as the *principle of stationary potential energy*. If at a stationary point the potential energy actually has a minimum value, the equilibrium state is called stable. In structural mechanics, these principles are of fundamental importance and form the basis for numerical methods of structural analysis.

To demonstrate the principle, we consider the symmetric two-bar truss shown in Fig. 9-8. The structure is subjected to a load W at node C. Under the action of this load, node C moves to a point C' . The problem is to compute the displacements x_1 and x_2 of node C. This can be done by writing the total potential energy of the structure in terms of x_1 and x_2 and then minimizing it. Once displacements x_1 and x_2 are known, member forces and stresses can be calculated using them. Let

E = modulus of elasticity, N/m^2 (this is the property of a material which relates stresses in the material to strains)

s = span of the truss, m

h = height of the truss, m

A_1 = cross-sectional area of member 1, m^2

A_2 = cross-sectional area of member 2, m^2

θ = angle at which load W is applied, degrees

L = length of the members; $L^2 = h^2 + 0.25s^2$, m

W = load, N

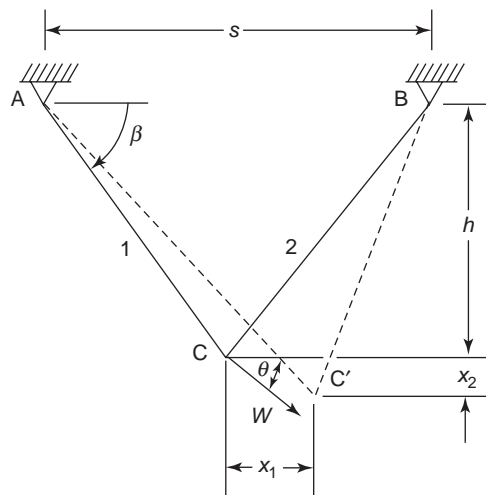


FIGURE 9-8 Two-bar truss.

x_1 = horizontal displacement, m
 x_2 = vertical displacement, m

The total potential energy of the system, assuming small displacements, is given as

$$P(x_1, x_2) = \frac{EA_1}{2L}(x_1 \cos \beta + x_2 \sin \beta)^2 + \frac{EA_2}{2L}(-x_1 \cos \beta + x_2 \sin \beta)^2 - Wx_1 \cos \theta - Wx_2 \sin \theta, \quad \text{N} \cdot \text{m} \quad (\text{a})$$

where the angle β is shown in Fig. 9-8. Minimization of P with respect to x_1 and x_2 gives the displacements x_1 and x_2 for the equilibrium state of the two-bar structure. Example 9.11 demonstrates this calculation.

EXAMPLE 9.11 Minimization of Total Potential Energy of a Two-Bar Truss

For the two-bar truss problem use the following numerical data: $A_1 = A_2 = 1.0\text{E-}05 \text{ m}^2$, $h = 1.0 \text{ m}$, $s = 1.5 \text{ m}$, $W = 10 \text{ kN}$, $\theta = 30^\circ$, $E = 207 \text{ GPa}$. Minimize the total potential energy given in Eq. (a) by (i) the graphical method, (ii) the analytical method, and (iii) the conjugate gradient method.

Solution. Substituting these data into Eq. (a) and simplifying, we get (note that $\cos \beta = s/2L$ and $\sin \beta = h/L$):

$$P(x_1, x_2) = \frac{EA}{L} \left(\frac{s}{2L} \right)^2 x_1^2 + \frac{EA}{L} \left(\frac{h}{2L} \right)^2 x_2^2 - Wx_1 \cos \theta - Wx_2 \sin \theta$$

$$= (5.962\text{E} + 06)x_1^2 + (1.0598\text{E} + 06)x_2^2 - 8660x_1 - 5000x_2, \quad \text{N} \cdot \text{m} \quad (\text{b})$$

Contours for the function are shown in Fig. 9-9. The optimum solution from the graph is read as $x_1 = (7.2634\text{E-}03)\text{m}$; $x_2 = (2.3359\text{E-}03)\text{m}$; $P = -37.348 \text{ N} \cdot \text{m}$. Using the necessary conditions of optimality ($\nabla P = \mathbf{0}$), we get

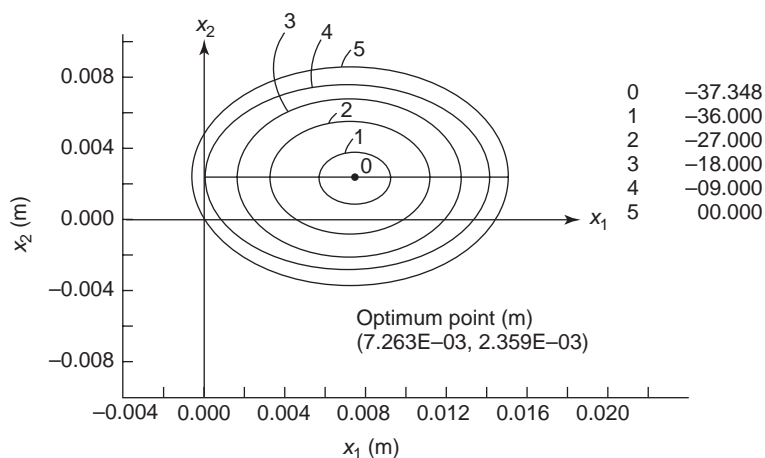


FIGURE 9-9 Contours of the potential energy function $P(x_1, x_2)$ for a two-bar truss ($P = 0, -9.0, -18.0, -27.0, -36.0, \text{ and } -37.348 \text{ N} \cdot \text{m}$).

$$2(5.962\text{E}+06)x_1 - 8660 = 0, \quad x_1 = (7.2629\text{E}-03), \text{m}$$

$$2(1.0598\text{E}+06)x_2 - 5000 = 0, \quad x_2 = (2.3589\text{E}-03), \text{m} \quad (\text{c})$$

The conjugate gradient method given in IDESIGN also converges to the same solution.

9.6.2 Solution of Nonlinear Equations

Unconstrained optimization methods can be used to find roots of a nonlinear system of equations. To demonstrate this, we consider the following 2×2 system:

$$F_1(x_1, x_2) = 0 \quad F_2(x_1, x_2) = 0 \quad (\text{a})$$

We define a function that is the sum of the squares of the functions F_1 and F_2 as

$$f(x_1, x_2) = F_1^2(x_1, x_2) + F_2^2(x_1, x_2) \quad (\text{b})$$

Note that if x_1 and x_2 are roots of Eq. (a), then $f = 0$ in Eq. (b). If x_1 and x_2 are not roots, then the function $f > 0$ represents the sum of the squares of the errors in the equations $F_1 = 0$ and $F_2 = 0$. Thus, the optimization problem is to find x_1 and x_2 to minimize the function $f(x_1, x_2)$ of Eq. (b). We need to show that the necessary conditions for minimization of $f(\mathbf{x})$ give roots for the nonlinear system of equations. The necessary conditions of optimality give

$$\frac{\partial f}{\partial x_1} = 2F_1 \frac{\partial F_1}{\partial x_1} + 2F_2 \frac{\partial F_2}{\partial x_1} = 0 \quad (\text{c})$$

$$\frac{\partial f}{\partial x_2} = 2F_1 \frac{\partial F_1}{\partial x_2} + 2F_2 \frac{\partial F_2}{\partial x_2} = 0 \quad (\text{d})$$

Note that the necessary conditions are satisfied if $F_1 = F_2 = 0$, i.e., x_1 and x_2 are roots of the equations $F_1 = 0$ and $F_2 = 0$. At this point $f = 0$. Note also that the necessary conditions can be satisfied if $\partial F_i / \partial x_j = 0$ for $i, j = 1, 2$. If $\partial F_i / \partial x_j = 0$, x_1 and x_2 are stationary points for the functions F_1 and F_2 . For most problems it is unlikely that stationary points for F_1 and F_2 will also be roots of $F_1 = 0$ and $F_2 = 0$, so we may exclude these cases. In any case, if x_1 and x_2 are roots of the equations, then f must have zero value. Also if the optimum value of f is different from zero ($f \neq 0$), then x_1 and x_2 cannot be roots of the nonlinear system. Thus, if the optimization algorithm converges with $f \neq 0$, then the optimum point for the problem of minimization of f is not a root of the nonlinear system. The algorithm should be restarted from a different point. Example 9.12 illustrates this root finding process.

EXAMPLE 9.12 Roots of Nonlinear Equations by Unconstrained Minimization

Find roots of the equations $F_1(\mathbf{x}) = 3x_1^2 + 12x_2^2 + 10x_1 = 0$; $F_2(\mathbf{x}) = 24x_1x_2 + 4x_2 + 3 = 0$.

Solution. We define the error function $f(\mathbf{x})$ as

$$f(\mathbf{x}) = F_1^2 + F_2^2 = (3x_1^2 + 12x_2^2 + 10x_1)^2 + (24x_1x_2 + 4x_2 + 3)^2 \quad (\text{a})$$

TABLE 9-4 Root of Nonlinear Equations of Example 9.12: The Conjugate Gradient Method

No.	x_1	x_2	F_1	F_2	f
0	-1.0000	1.0000	5.0000	-17.0000	314.0000
1	-0.5487	0.4649	-1.9900	-1.2626	5.5530
2	-0.4147	0.5658	0.1932	-0.3993	0.1968
3	-0.3993	0.5393	-0.0245	-0.0110	7.242E-4
4	-0.3979	0.5403	-9.377E-4	-1.550E-3	2.759E-6
5	-0.3980	0.5404	-4.021E-4	-3.008E-4	1.173E-8

To minimize this function, we can use any of the methods discussed previously. Table 9-4 shows the iteration history with the conjugate gradient method available in IDESIGN (Arora and Tseng, 1987a,b). A root of the equations is $x_1 = -0.3980$, $x_2 = 0.5404$ starting from the point $(-1, 1)$. Starting from the point $(-50, 50)$ another root is found as $(-3.331, 0.03948)$. However, starting from another point $(2, 3)$, the program converges to $(0.02063, -0.2812)$ with $f = 4.351$. Since $f \neq 0$, this point is not a root of the given system of equations. When this happens, we start from a different point and re-solve the problem.

Note that the preceding procedure can be generalized to a system of n equations in n unknowns. In this case, the error function $f(\mathbf{x})$ will be defined as

$$f(\mathbf{x}) = \sum_{i=1}^n [F_i(\mathbf{x})]^2 \quad (\text{b})$$

9.7 Solution of Constrained Problems Using Unconstrained Optimization Methods

It turns out that unconstrained optimization methods can also be used to solve constrained design problems. This section briefly describes such methods that transform the constrained problem to a sequence of unconstrained problems. The basic idea is to construct a composite function using the cost and constraint functions. It also contains certain parameters—called the penalty parameters—that penalize the composite function for violation of constraints. The larger the violation, the larger the penalty. Once the composite function is defined for a set of penalty parameters, it is minimized using any of the unconstrained optimization techniques. The penalty parameters are then adjusted based on certain conditions, and the composite function is redefined and minimized. The process is continued until there is no significant improvement in the estimate for the optimum point.

Methods based on the foregoing philosophy have been generally called sequential unconstrained minimization techniques, or in short SUMT (Fiacco and McCormick, 1968). It is seen that the basic idea of SUMT is quite straightforward. Because of their simplicity, the methods have been extensively developed and tested for engineering design problems. A very brief discussion of the basic concepts and philosophy of the methods is included in the text to give the students a flavor for the techniques. For more detailed presentations, texts by Gill and coworkers (1981), Reklaitis and coworkers (1983), and others should be consulted.

The term “transformation method” is used to describe any method that solves the constrained optimization problem by transforming it into one or more unconstrained problems. They include the so-called penalty and barrier function methods (exterior and interior penalty

methods, respectively) as well as the *multiplier methods* (also called *augmented Lagrangian methods*). To remind the reader of the original constrained problem that we are trying to solve, we re-state it as follows: Find an n -vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$ to minimize a cost function $f = f(\mathbf{x})$ subject to $h_i(\mathbf{x}) = 0$; $i = 1$ to p and $g_i(\mathbf{x}) \leq 0$; $i = 1$ to m . All transformation methods convert this constrained optimization problem into an unconstrained problem using a *transformation function* of the form:

$$\phi(\mathbf{x}, \mathbf{r}) = f(\mathbf{x}) + P(\mathbf{h}(\mathbf{x}), \mathbf{g}(\mathbf{x}), \mathbf{r}) \quad (9.14)$$

where \mathbf{r} is a vector of penalty parameters and P is a real valued function whose action of imposing the penalty on the cost function is controlled by \mathbf{r} . The form of penalty function P depends on the method used. The basic procedure is to choose an initial design estimate $\mathbf{x}^{(0)}$ and define the function ϕ of Eq. (9.14). The penalty parameters \mathbf{r} are also initially selected. The function ϕ is minimized for \mathbf{x} , keeping \mathbf{r} fixed. Then the parameters \mathbf{r} are adjusted and the procedure is repeated until no further improvement is possible.

9.7.1 Sequential Unconstrained Minimization Techniques

Sequential unconstrained minimization techniques consist of two different types of penalty functions. The first one is called the *penalty function* method and the second is called the *barrier function* method. The basic idea of the penalty function approach is to define the function P in Eq. (9.14) in such a way that if there are constraint violations, the cost function $f(\mathbf{x})$ gets penalized by addition of a positive value. Several penalty functions can be defined. The most popular one is called the *quadratic loss function* defined as

$$P(\mathbf{h}(\mathbf{x}), \mathbf{g}(\mathbf{x}), r) = r \left\{ \sum_{i=1}^p [h_i(\mathbf{x})]^2 + \sum_{i=1}^m [g_i^+(\mathbf{x})]^2 \right\} \quad (9.15)$$

where $g_i^+(\mathbf{x}) = \max(0, g_i(\mathbf{x}))$, and r is a scalar penalty parameter. Note that $g_i^+(\mathbf{x}) \geq 0$; it is zero if the inequality is inactive ($g_i(\mathbf{x}) < 0$) and it is positive if the inequality is violated. It can be seen that if the equality constraint is not satisfied, i.e., $h_i(\mathbf{x}) \neq 0$, or the inequality is violated, i.e., $g_i(\mathbf{x}) > 0$, then Eq. (9.15) gives a positive value to the function P , and the cost function is penalized, as seen in Eq. (9.14). The starting design point for the method can be arbitrary. The methods based on the philosophy of penalty functions are sometimes called the *exterior methods* because they iterate through the infeasible region.

The advantages and disadvantages of the penalty function method are:

1. It is applicable to general constrained problems with equality and inequality constraints.
2. The starting design point can be arbitrary.
3. The method iterates through the infeasible region where the problem functions may be undefined.
4. If the iterative process terminates prematurely, the final design may not be feasible and hence not usable.

The *barrier function methods* are applicable only to the inequality constrained problems. Popular barrier functions are:

1. Inverse barrier function

$$P(\mathbf{g}(\mathbf{x}), r) = \frac{1}{r} \sum_{i=1}^m \frac{-1}{g_i(\mathbf{x})} \quad (9.16)$$

2. Log barrier function

$$P(\mathbf{g}(\mathbf{x}), r) = \frac{1}{r} \sum_{i=1}^m \log(-g_i(\mathbf{x})) \quad (9.17)$$

These are called the barrier function methods because a large barrier is constructed around the feasible region. In fact, the function P becomes infinite if any of the inequalities is active. Thus, when the iterative process is started from a feasible point, it cannot go into the infeasible region because the iterative process cannot cross the huge barrier. For both penalty function and barrier function methods, it can be shown that as $r \rightarrow \infty$, $\mathbf{x}(r) \rightarrow \mathbf{x}^*$, where $\mathbf{x}(r)$ is a point that minimizes the transformed function $\phi(\mathbf{x}, r)$ of Eq. (9.14) and \mathbf{x}^* is a solution of the original constrained optimization problem.

The advantages and disadvantages of the barrier function method are:

1. The method is applicable to inequality constrained problems only.
2. The starting design point must be feasible. It turns out, however, that the method itself can be used to determine the starting point (Haug and Arora, 1979).
3. The method always iterates through the feasible region, so if it terminates prematurely, the final design is feasible and hence usable.

The sequential unconstrained minimization techniques have certain weaknesses that are most serious when r is large. The penalty and barrier functions tend to be ill-behaved near the boundary of the feasible set where the optimum points usually lie. There is also a problem of selecting the sequence $r^{(k)}$. The choice of $r^{(0)}$ and the rate at which $r^{(k)}$ tends to infinity can seriously affect the computational effort to find a solution. Furthermore, the Hessian matrix of the unconstrained function becomes ill-conditioned as $r \rightarrow \infty$.

9.7.2 Multiplier (Augmented Lagrangian) Methods

To alleviate some of the difficulties of the methods presented in the previous section, a different class of transformation methods has been developed in the literature. These are called the *multiplier or augmented Lagrangian methods*. In these methods, there is no need for the penalty parameters r to go to infinity. As a result the transformation function ϕ has good conditioning with no singularities. The multiplier methods are convergent as are the SUMTs. That is, they converge to a local minimum starting from any point. It has been proven that they possess a faster rate of convergence than the two methods of the previous subsection. With multiplier methods, the penalty function is given as

$$P(\mathbf{h}(\mathbf{x}), \mathbf{g}(\mathbf{x}), \mathbf{r}, \boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^p r_i (h_i + \theta_i')^2 + \frac{1}{2} \sum_{i=1}^m r_i [(g_i + \theta_i)^+]^2 \quad (9.18)$$

where $\theta_i > 0$, $r_i > 0$, and θ_i, r_i' are parameters associated with the i th inequality and equality constraints. If $\theta_i = \theta_i' = 0$ and $r_i = r_i' = r$, then Eq. (9.18) reduces to the well-known quadratic loss function given in Eq. (9.15), where convergence is enforced by letting $r \rightarrow \infty$. However, the objective of the multiplier methods is to keep each r_i and r_i' finite. The idea of multiplier methods is to start with some r_i, r_i', θ_i' , and θ_i and minimize the transformation function of Eq. (9.14). The parameters r_i, r_i', θ_i' , and θ_i are then adjusted using some procedures and the entire process is repeated until optimality conditions are satisfied. For a more detailed discussion and applications of the methods, Arora and coworkers (1991) and Arora (1999) and the references cited in there may be consulted.

Exercises for Chapter 9*

Section 9.1 More on Step Size Determination

- 9.1 Write a computer program to implement the polynomial interpolation with a quadratic curve fitting. Choose a function $f(\alpha) = 7\alpha^2 - 20\alpha + 22$. Use the golden section method to initially bracket the minimum point of $f(\alpha)$ with $\delta = 0.05$. Use your program to find the minimum point of $f(\alpha)$. Comment on the accuracy of the solution.
- 9.2 For the function $f(\alpha) = 7\alpha^2 - 20\alpha + 22$, use two function values, $f(0)$ and $f(\alpha_u)$, and the slope of f at $\alpha = 0$ to fit a quadratic curve. Here α_u is any upper bound on the minimum point of $f(\alpha)$. What is the estimate of the minimum point from the preceding quadratic curve? How many iterations will be required to find α^* ? Why?
- 9.3 Under what situation can the polynomial interpolation approach not be used for one-dimensional minimization?
- 9.4 Given

$$f(\mathbf{x}) = 10 - x_1 + x_1x_2 + x_2^2$$

$$\mathbf{x}^{(0)} = (2, 4); \quad \mathbf{d}^{(0)} = (-1, -1)$$

For the one-dimensional search, three values of α , $\alpha_i = 0$, $\alpha_i = 2$, and $\alpha_u = 4$ are tried. Using quadratic polynomial interpolation, determine

1. At what value of α is the function a minimum? Prove that this is a minimum point and not a maximum.
2. At what values of α is $f(\alpha) = 15$?

Section 9.2 More on Steepest Descent Method

Verify the properties of the gradient vector for the following functions at the given point.

9.5 $f(\mathbf{x}) = 6x_1^2 - 6x_1x_2 + 2x_2^2 - 5x_1 + 4x_2 + 2$; $\mathbf{x}^{(0)} = (-1, -2)$

9.6 $f(\mathbf{x}) = 3x_1^2 + 2x_1x_2 + 2x_2^2 + 7$; $\mathbf{x}^{(0)} = (5, 10)$

9.7 $f(\mathbf{x}) = 10(x_1^2 - x_2) + x_1^2 - 2x_1 + 5$; $\mathbf{x}^{(0)} = (-1, 3)$

Section 9.3 Scaling of Design Variables

- 9.8 Consider the following three functions:

$$f_1 = x_1^2 + x_2^2 + x_3^2; \quad f_2 = x_1^2 + 10x_2^2 + 100x_3^2; \quad f_3 = 100x_1^2 + x_2^2 + 0.1x_3^2$$

Minimize f_1 , f_2 , and f_3 using the program for the steepest descent method given in Appendix D. Choose the starting design to be $(1, 1, 2)$ for all functions. What do you conclude from observing the performance of the method on the foregoing functions? How would you scale the design variables for the functions f_2 and f_3 to improve the rate of convergence of the method?

Section 9.4 Search Direction Determination: Newton's Method

- 9.9 Answer True or False.

1. In Newton's method, it is always possible to calculate a search direction at any point.

2. The Newton direction is always that of descent for the cost function.
3. Newton's method is convergent starting from any point with a step size of 1.
4. Newton's method needs only gradient information at any point.

For the following problems, complete one iteration of the modified Newton's method; also check the descent condition for the search direction.

- | | |
|--------------------|--------------------|
| 9.10 Exercise 8.52 | 9.11 Exercise 8.53 |
| 9.12 Exercise 8.54 | 9.13 Exercise 8.55 |
| 9.14 Exercise 8.56 | 9.15 Exercise 8.57 |
| 9.16 Exercise 8.58 | 9.17 Exercise 8.59 |
| 9.18 Exercise 8.60 | 9.19 Exercise 8.61 |
- 9.20 Write a computer program to implement the modified Newton's algorithm. Use equal interval search for line search. Solve Exercises 8.52 to 8.61 using the program.

Section 9.5 Search Direction Determination: Quasi-Newton Methods

- 9.21 *Answer True or False for unconstrained problems.*
1. The DFP method generates an approximation to the inverse of the Hessian.
 2. The DFP method generates a positive definite approximation to the inverse of the Hessian.
 3. The DFP method always gives a direction of descent for the cost function.
 4. The BFGS method generates a positive definite approximation to the Hessian of the cost function.
 5. The BFGS method always gives a direction of descent for the cost function.
 6. The BFGS method always converges to the Hessian of the cost function.

For the following problems, complete two iterations of the Davidon-Fletcher-Powell method.

- | | |
|--------------------|--------------------|
| 9.22 Exercise 8.52 | 9.23 Exercise 8.53 |
| 9.24 Exercise 8.54 | 9.25 Exercise 8.55 |
| 9.26 Exercise 8.56 | 9.27 Exercise 8.57 |
| 9.28 Exercise 8.58 | 9.29 Exercise 8.59 |
| 9.30 Exercise 8.60 | 9.31 Exercise 8.61 |
- 9.32 Write a computer program to implement the Davidon-Fletcher-Powell method. Solve Exercises 8.52 to 8.61 using the program.

For the following problems, complete two iterations of the BFGS method.

- | | |
|--------------------|--------------------|
| 9.33 Exercise 8.52 | 9.34 Exercise 8.53 |
| 9.35 Exercise 8.54 | 9.36 Exercise 8.55 |
| 9.37 Exercise 8.56 | 9.38 Exercise 8.57 |
| 9.39 Exercise 8.58 | 9.40 Exercise 8.59 |
| 9.41 Exercise 8.60 | 9.42 Exercise 8.61 |
- 9.43 Write a computer program to implement the BFGS method. Solve Exercises 8.52 to 8.61 using the program.

Section 9.6 Engineering Applications of Unconstrained Methods

Find the equilibrium configuration for the two-bar structure of Fig. 9-8 using the following numerical data.

$$9.44 \quad A_1 = 1.5 \text{ cm}^2, A_2 = 2.0 \text{ cm}^2, h = 100 \text{ cm}, s = 150 \text{ cm}, W = 100,000 \text{ N}, \theta = 45^\circ, E = 21 \text{ MN/cm}^2$$

$$9.45 \quad A_1 = 100 \text{ mm}^2, A_2 = 200 \text{ mm}^2, h = 1000 \text{ mm}, s = 1500 \text{ mm}, W = 50,000 \text{ N}, \theta = 60^\circ, E = 210,000 \text{ N/mm}^2$$

Find roots of the following nonlinear equations using the conjugate gradient method.

$$9.46 \quad F(x) = 3x - e^x = 0$$

$$9.47 \quad F(x) = \sin x = 0$$

$$9.48 \quad F(x) = \cos x = 0$$

$$9.49 \quad F(x) = \frac{2x}{3} - \sin x = 0$$

$$9.50 \quad F_1(\mathbf{x}) = 1 - \frac{10}{x_1^2 x_2} = 0, \quad F_2(\mathbf{x}) = 1 - \frac{2}{x_1 x_2^2} = 0$$

$$9.51 \quad F_1(\mathbf{x}) = 5 - \frac{1}{8} x_1 x_2 - \frac{1}{4 x_1^2} x_2^2 = 0, \quad F_2(\mathbf{x}) = -\frac{1}{16} x_1^2 + \frac{1}{2 x_1} x_2 = 0$$

10 Numerical Methods for Constrained Optimum Design

Upon completion of this chapter, you will be able to:

- Explain basic steps of a numerical algorithm for solution of constrained optimization problems
- Explain the concepts of a descent direction and descent step for the constrained nonlinear optimization problem
- Linearize the constrained nonlinear optimization problem and define a linear programming subproblem
- Use sequential linear programming algorithm to solve constrained nonlinear optimization problems
- Define a quadratic programming subproblem for the constrained nonlinear optimization problem
- Use an optimization algorithm to solve nonlinear optimization problems
- Use Excel Solver to optimize some engineering design problems

In the previous chapter, the constrained nonlinear programming problem was transformed into a sequence of unconstrained problems. In this chapter, we describe numerical methods—sometimes called the *primal methods*—to directly solve the original constrained problem. For convenience of reference, the problem defined in Section 2.11 is restated as: find $\mathbf{x} = (x_1, \dots, x_n)$, a design variable vector of dimension n , to minimize a cost function $f = f(\mathbf{x})$ subject to equality and inequality constraints

$$h_i(\mathbf{x}) = 0, \quad i = 1 \text{ to } p; \quad g_i(\mathbf{x}) \leq 0, \quad i = 1 \text{ to } m \quad (10.1)$$

and the explicit bounds on design variables $x_{il} \leq x_i \leq x_{iu}$; $i = 1$ to n , where x_{il} and x_{iu} are, respectively, the smallest and largest allowed values for the i th design variable x_i . Note that these constraints are quite simple and easy to treat in actual numerical implementations. It is usually efficient to treat them in that manner. However, in the discussion and illustration of the numerical methods, we shall assume that they are included in the inequality constraints in Eq. (10.1). Note also that we shall present only the methods that can treat the general

constrained problem with equality and inequality constraints defined in Eq. (10.1). That is, the methods that treat only equalities or only inequalities will not be presented.

Just as for unconstrained problems, several methods have been investigated for the preceding model of general constrained optimization problems. Most methods follow the two-phase approach as before: *search direction* and *step size* determination phases. The approach followed here will be to describe the underlying *ideas* and *concepts* of the methods. A comprehensive coverage of all the methods giving their advantages and disadvantages will be avoided. Only a few simple and generally applicable methods will be described and illustrated with examples.

In Section 8.3 we described the steepest descent method for solving unconstrained optimization problems. That method is quite straightforward. It is, however, not directly applicable to constrained problems. One reason is that we must consider constraints while computing the search direction and the step size. In this chapter, we shall describe a *constrained steepest descent method* that computes the direction of design change considering local behavior of cost and constraint functions. The method (and most others) is based on linearization of the problem about the current estimate of the optimum design. Therefore, linearization of the problem is quite important and is discussed in detail. Once the problem has been linearized, it is natural to ask if it can be solved using linear programming methods. Therefore, we shall first describe a method that is a simple extension of the Simplex method of linear programming. Then we shall discuss extension of the steepest descent method to constrained problems.

10.1 Basic Concepts and Ideas

This section contains basic concepts, ideas, and definitions of the terms used in numerical methods for constrained optimization. The status of a constraint at a design point is defined. Active, inactive, violated, and ε -active constraints are defined. Normalization of constraints and its advantages are explained with examples. The ideas of a descent function and convergence of algorithms are explained.

10.1.1 Basic Concepts Related to Algorithms for Constrained Problems

In the *direct numerical (search) methods*, we select a design to initiate the iterative process, as for the unconstrained methods described in Chapter 8. The iterative process is continued until no further moves are possible and the optimality conditions are satisfied. Most of the general concepts of iterative numerical algorithms discussed in Section 8.1 also apply to methods for constrained optimization problems. Therefore, those concepts should be thoroughly reviewed.

All numerical methods discussed in this chapter are based on the following iterative prescription as also given in Eqs. (8.1) and (8.2) for unconstrained problems:

$$\text{vector form:} \quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}; \quad k = 0, 1, 2, \dots \quad (10.2)$$

$$\text{component form:} \quad x_i^{(k+1)} = x_i^{(k)} + \Delta x_i^{(k)}; \quad k = 0, 1, 2, \dots; \quad i = 1 \text{ to } n \quad (10.3)$$

The superscript k represents the iteration or design cycle number, subscript i refers to the i th design variable, $\mathbf{x}^{(0)}$ is the starting design estimate, and $\Delta\mathbf{x}^{(k)}$ represents a change in the current design. As in the unconstrained numerical methods, the change in design $\Delta\mathbf{x}^{(k)}$ is decomposed as

$$\Delta\mathbf{x}^{(k)} = \alpha_k \mathbf{d}^{(k)} \quad (10.4)$$

where α_k is a step size in the search direction $\mathbf{d}^{(k)}$. Thus, the design improvement involves the solution of the search direction and step size determination subproblems. Solution of both the subproblems can involve values of cost and constraint functions as well as their gradients at the current design point.

Conceptually, algorithms for unconstrained and constrained optimization problems are based on the same iterative philosophy. There is one important difference, however; constraints must be considered while determining the search direction as well as the step size for the constrained problems. A different procedure for determining either one can give a different optimization algorithm. We shall describe, in general terms, a couple of ways in which the algorithms may proceed in the design space. All algorithms need a design estimate to initiate the iterative process. The starting design can be feasible or infeasible. If it is inside the feasible set as Point A in Fig. 10-1, then there are two possibilities:

1. The gradient of the cost function vanishes at the point so it is an unconstrained stationary point. We need to check the sufficient condition for optimality of the point.
2. If the current point is not stationary, then we can reduce the cost function by moving along a descent direction, say, the steepest descent direction ($-c$) as shown in Fig. 10-1. We continue such iterations until either a constraint is encountered or an unconstrained minimum point is reached.

For the remaining discussion, we assume that the optimum point is on the boundary of the feasible set, i.e., some constraints are active. Once the constraint boundary is encountered at Point B, one strategy is to travel along a tangent to the boundary such as the direction B–C in Fig. 10-1. This results in an infeasible point from where the constraints are corrected to again reach the feasible point D. From there the preceding steps are repeated until the optimum point is reached. Another strategy is to deflect the tangential direction B–C toward the feasible region by a certain angle θ . Then a line search is performed through the feasible region to reach the boundary point E, as shown in Fig. 10-1. The procedure is then repeated from there.

When the starting point is infeasible, as Point A in Fig. 10-2, then one strategy is to correct constraints to reach the constraint boundary at Point B. From there, the strategies described in the preceding paragraph can be followed to reach the optimum point. This is shown in Path 1 in Fig. 10-2. The second strategy is to iterate through the infeasible region by computing directions that take successive design points closer to the optimum point, shown as Path 2 in Fig. 10-2.

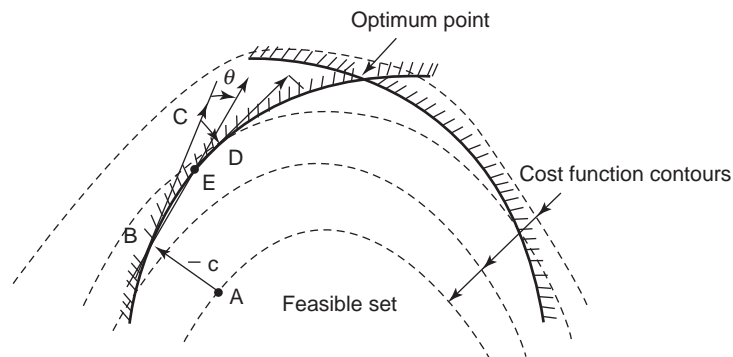


FIGURE 10-1 Conceptual steps of constrained optimization algorithms initiated from a feasible point.

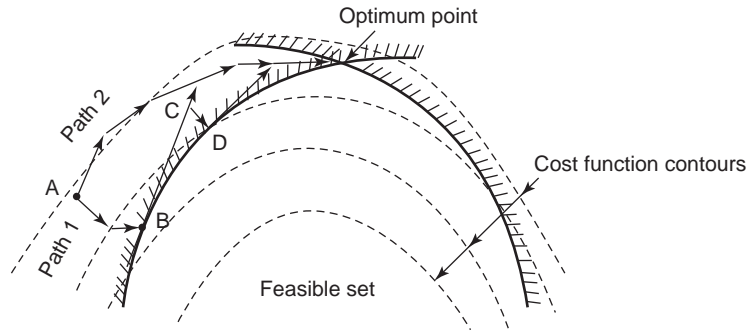


FIGURE 10-2 Conceptual steps of constrained optimization algorithms initiated from an infeasible point.

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. A few algorithms work well if the problem has only inequality constraints whereas others can treat both equality and inequality constraints simultaneously. In this text, we shall concentrate mostly on general algorithms that have no restriction on the form of the functions or the constraints. Most of the algorithms that we shall describe will be based on the following *four basic steps*:

1. *Linearization of cost and constraint functions* about the current design point.
2. *Definition of a search direction determination subproblem* using the linearized functions.
3. *Solution of the subproblem* that gives a search direction in the design space.
4. Calculation of a *step size* to minimize a descent function in the search direction.

10.1.2 Constraint Status at a Design Point

An inequality constraint can be either active, ε -active, violated, or inactive at a design point. On the other hand, an equality constraint is either active or violated at a design point. The precise definitions of the status of a constraint at a design point are needed in the development and discussion of numerical methods.

Active Constraint An inequality constraint $g_i(\mathbf{x}) \leq 0$ is said to be *active* (or *tight*) at a design point $\mathbf{x}^{(k)}$ if it is satisfied as an equality at that point, i.e., $g_i(\mathbf{x}^{(k)}) = 0$.

Inactive Constraint An inequality constraint $g_i(\mathbf{x}) \leq 0$ is said to be *inactive* at a design point $\mathbf{x}^{(k)}$ if it has negative value at that point, i.e., $g_i(\mathbf{x}^{(k)}) < 0$.

Violated Constraint An *inequality* constraint $g_i(\mathbf{x}) \leq 0$ is said to be *violated* at a design point $\mathbf{x}^{(k)}$ if it has positive value there, i.e., $g_i(\mathbf{x}^{(k)}) > 0$. An *equality* constraint $h_i(\mathbf{x}^{(k)}) = 0$ is *violated* at a design point $\mathbf{x}^{(k)}$ if it has nonzero value there, i.e., $h_i(\mathbf{x}^{(k)}) \neq 0$. Note that by these definitions, an equality constraint is always either active or violated for any design point.

ε -Active Constraint Any inequality constraint $g_i(\mathbf{x}^{(k)}) \leq 0$ is said to be ε -active at the point $\mathbf{x}^{(k)}$ if $g_i(\mathbf{x}^{(k)}) < 0$ but $g_i(\mathbf{x}^{(k)}) + \varepsilon \geq 0$, where $\varepsilon > 0$ is a small number. This means that the point is close to the constraint boundary on the feasible side (within an ε -band as shown in Fig. 10-3).

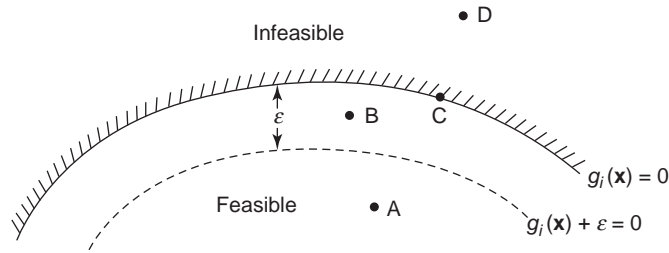


FIGURE 10-3 Status of a constraint at design points A, B, C, and D.

To understand the idea of the status of a constraint, we refer to Fig. 10-3. Consider the i th inequality constraint $g_i(\mathbf{x}) \leq 0$. The constraint boundary (surface in n -dimensional space), $g_i(\mathbf{x}) = 0$, is plotted, and feasible and infeasible sides for the constraint are identified. An artificial boundary at a distance of ε from the boundary $g_i(\mathbf{x}) = 0$ and inside the feasible region is also plotted. We consider four design points A, B, C, and D as shown in Fig. 10-3. For design point A, the constraint $g_i(\mathbf{x})$ is negative and even $g_i(\mathbf{x}) + \varepsilon < 0$. Thus, the constraint is *inactive* for design point A. For design point B, $g_i(\mathbf{x})$ is strictly less than zero, so it is inactive. However, $g_i(\mathbf{x}) + \varepsilon > 0$, so the constraint is ε -*active* for design point B. For design point C, $g_i(\mathbf{x}) = 0$ as shown in Fig. 10-3. Therefore, the constraint is *active* there. For design point D, $g_i(\mathbf{x})$ is greater than zero, so the constraint is *violated*.

10.1.3 Constraint Normalization

In numerical calculations, it is desirable to normalize all the constraint functions. As noted earlier, active and violated constraints are used in computing a desirable direction of design change. Usually one value for ε (say 0.10) is used for all constraints. Since different constraints involve different orders of magnitude, it is not proper to use the same ε for all the constraints unless they are normalized. For example, consider a stress constraint as

$$\sigma \leq \sigma_a, \quad \text{or} \quad \sigma - \sigma_a \leq 0 \quad (10.5)$$

and a displacement constraint as

$$\delta \leq \delta_a, \quad \text{or} \quad \delta - \delta_a \leq 0 \quad (10.6)$$

where

- σ = calculated stress at a point
- σ_a = an allowable stress
- δ = calculated deflection at a point
- δ_a = an allowable deflection

Note that the units for the two constraints are different. Constraint of Eq. (10.5) involves stress, which has units of Pascals (Pa, N/m²). For example, allowable stress for steel is 250 MPa. The other constraint in Eq. (10.6) involves deflections of the structure, which may be only a few centimeters. For example, allowable deflection δ_a may be only 2 cm. Thus, the values of the two constraints are of widely differing orders of magnitude. If the constraints are violated, it is difficult to judge the severity of their violation. We can, however, normalize the constraints by dividing them by their respective allowable values to obtain the normalized constraint as

$$R - 1.0 \leq 0 \quad (10.7)$$

where $R = \sigma/\sigma_a$ for the stress constraint and $R = \delta/\delta_a$ for the deflection constraint. Here, both σ_a and δ_a are assumed to be positive; otherwise, the sense of the inequality changes. For normalized constraints, it is easy to check for ε -active constraint using the same value of ε for both of them.

There are other constraints that must be written in the form

$$1.0 - R \leq 0 \quad (10.8)$$

when normalized with respect to their nominal value. For example, the fundamental vibration frequency ω of a structure or a structural element must be above a given threshold value of ω_a , i.e., $\omega \geq \omega_a$. When the constraint is normalized and converted to the standard "less than" form, it is given as in Eq. (10.8) with $R = \omega/\omega_a$. In subsequent discussions, it is assumed that all equality as well as inequality constraints have been converted to the normalized form.

There are some constraints that cannot be normalized. For these constraints the allowable values may be zero. For example, the lower bound on some design variables may be zero. Such constraints cannot be normalized with respect to lower bounds. These constraints may be kept in the original form. Or, they may be divided by 100 to transform them to a percent value. Example 10.1 illustrates the constraint normalization process and checking of the constraint status.

EXAMPLE 10.1 Constraint Normalization and Status at a Point

Consider the two constraints:

$$\bar{h} = x_1^2 + \frac{1}{2}x_2 = 18 \quad (a)$$

$$\bar{g} = 500x_1 - 30,000x_2 \leq 0 \quad (b)$$

At the design points (1, 1) and (-4.5, -4.5), investigate whether the constraints are active, violated, ε -active, or inactive. Use $\varepsilon = 0.1$ to check ε -active constraints.

Solution. Let us normalize the constraint \bar{h} and express it in the standard form as

$$h = \frac{1}{18}x_1^2 + \frac{1}{36}x_2 - 1.0 = 0 \quad (c)$$

Evaluating the constraint at the two points we get, $h(1, 1) = -0.9166$, and $h(-4.5, -4.5) = 0$. Therefore, the equality constraint is violated at (1, 1) and active at (-4.5, -4.5).

The inequality constraint \bar{g} cannot be normalized by dividing it by $500x_1$ or $30,000x_2$ because x_1 and x_2 can have negative values which will change the sense of the inequality. We must normalize the constraint functions using only positive constants or positive variables. To treat this situation, we may divide the constraint by $30,000|x_2|$ and obtain a normalized constraint as $x_1/60|x_2| - x_2/|x_2| \leq 0$. This type of nor-

malization is, however, not desirable since it changes the nature of the constraint from linear to nonlinear. Linear constraints are more efficient to treat than the nonlinear constraints in numerical calculations. Therefore, care and judgment needs to be exercised while normalizing constraints. If a normalization procedure does not work, another procedure should be tried. In some cases, it may be better to use the constraints in their original form, especially the equality constraints. Thus, in numerical calculations, some experimentation with normalization of constraints may be needed for some forms of the constraints. For the present constraint, we normalize it with respect to the constant 500 and then divide by 100 to obtain it in the percent form as

$$g = \frac{1}{100}(x_1 - 60x_2) \leq 0 \quad (d)$$

At $(1, 1)$, $g = -0.59 < 0$ (so, inactive) and at $(-4.5, -4.5)$, $g = 2.655 > 0$ (so, violated).

10.1.4 Descent Function

For unconstrained optimization, each algorithm in Chapters 8 and 9 required reduction in the cost function at every design iteration. With that requirement, a descent toward the minimum point was maintained. *A function used to monitor progress toward the minimum is called the descent function or the merit function.* The cost function is used as the descent function in unconstrained optimization problems. *The idea of a descent function is very important in constrained optimization as well.* Use of the cost function, however, as a descent function for constrained optimization is quite cumbersome. Therefore, many other descent functions have been proposed and used. We shall discuss one such function later in this chapter. At this point, the purpose of the descent function should be well understood. The basic idea is to compute a search direction $\mathbf{d}^{(k)}$ and then a step size along it such that the descent function is reduced. With this requirement proper progress toward the minimum point is maintained. The descent function also has the property that its minimum value is the same as that of the original cost function.

10.1.5 Convergence of an Algorithm

The idea of convergence of an algorithm is very important in constrained optimization problems. We first define and then discuss its importance, and how to achieve it. *An algorithm is said to be convergent if it reaches a minimum point starting from an arbitrary point.* An algorithm that has been proven to converge starting from an arbitrary point is called a *robust* method. In practical applications of optimization, such *reliable algorithms* are highly desirable. Many engineering design problems require considerable numerical effort to evaluate functions and their gradients. Failure of the algorithm can have disastrous effects with respect to wastage of valuable resources as well as morale of designers. Thus, it is extremely important to develop convergent algorithms for practical applications. It is equally important to enforce convergence in numerical implementation of algorithms in general purpose design optimization software. *A convergent algorithm satisfies the following two requirements:*

1. There is a descent function for the algorithm. The idea is that the descent function must decrease at each iteration. This way, progress towards the minimum point can be monitored.

2. The direction of design change $\mathbf{d}^{(k)}$ is a continuous function of the design variables. This is also an important requirement. It implies that a proper direction can be found such that descent toward the minimum point can be maintained. This requirement also avoids “oscillations,” or “zigzagging” in the descent function.

In addition to the preceding two requirements, the assumption of the feasible set for the problem being closed and bounded must be satisfied for the guarantee of the algorithms to be convergent to a local minimum point. The algorithm may or may not converge if the two conditions are not satisfied. The feasible set is *closed* if all the boundary points are included in the set; i.e., there are no strict inequalities in the problem formulation. A *bounded set* implies that there are upper and lower bounds on the elements of the set. These requirements are satisfied if all functions of the problem are continuous. The preceding assumptions are not unreasonable for many engineering design applications.

10.2 Linearization of Constrained Problem

At each iteration, most numerical methods for constrained optimization compute design change by solving an approximate subproblem that is obtained by writing linear Taylor’s expansions for the cost and constraint functions. This idea of approximate or linearized subproblems is central to the development of many numerical optimization methods and should be thoroughly understood.

All search methods start with a design estimate and iteratively improve it. Let $\mathbf{x}^{(k)}$ be the design estimate at the k th iteration and $\Delta\mathbf{x}^{(k)}$ be the change in design. Writing Taylor’s expansion of the cost and constraint functions about the point $\mathbf{x}^{(k)}$, we obtain the linearized subproblem as

$$\text{minimize } f(\mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}) \cong f(\mathbf{x}^{(k)}) + \nabla f^T(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} \quad (10.9)$$

subject to the linearized equality constraints

$$h_j(\mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}) \cong h_j(\mathbf{x}^{(k)}) + \nabla h_j^T(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} = 0; \quad j = 1 \text{ to } p \quad (10.10)$$

and the linearized inequality constraints

$$g_j(\mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}) \cong g_j(\mathbf{x}^{(k)}) + \nabla g_j^T(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} \leq 0; \quad j = 1 \text{ to } m \quad (10.11)$$

where ∇f , ∇h_j and ∇g_j are gradients of the cost function, j th equality constraint, and j th inequality constraint, respectively, and “ \cong ” implies approximate equality. In the following presentation, we introduce some simplified notations for the current design $\mathbf{x}^{(k)}$ as follows:

Cost function value:

$$f_k = f(\mathbf{x}^{(k)}) \quad (10.12)$$

Negative of the j th equality constraint function value:

$$e_j = -h_j(\mathbf{x}^{(k)}) \quad (10.13)$$

Negative of the j th inequality constraint function value:

$$b_j = -g_j(\mathbf{x}^{(k)}) \quad (10.14)$$

Derivative of the cost function with respect to x_i :

$$c_i = \partial f(\mathbf{x}^{(k)}) / \partial x_i \quad (10.15)$$

Derivative of h_j with respect to x_i :

$$n_{ij} = \partial h_j(\mathbf{x}^{(k)}) / \partial x_i \quad (10.16)$$

Derivative of g_j with respect to x_i :

$$a_{ij} = \partial g_j(\mathbf{x}^{(k)}) / \partial x_i \quad (10.17)$$

Design change:

$$d_i = \Delta x_i^{(k)} \quad (10.18)$$

Note also that the linearization of the problem is done at any design iteration, so the argument $\mathbf{x}^{(k)}$ as well as the superscript k indicating the iteration number shall be omitted for some quantities. Using these notations, the approximate subproblem given in Eqs. (10.9) to (10.11) gets defined as follows:

$$\text{minimize } \bar{f} = \sum_{i=1}^n c_i d_i \quad (\bar{f} = \mathbf{c}^T \mathbf{d}) \quad (10.19)$$

subject to the linearized equality constraints:

$$\sum_{i=1}^n n_{ij} d_i = e_j; j = 1 \text{ to } p \quad (\mathbf{N}^T \mathbf{d} = \mathbf{e}) \quad (10.20)$$

and the linearized inequality constraints:

$$\sum_{i=1}^n a_{ij} d_i \leq b_j; j = 1 \text{ to } m \quad (\mathbf{A}^T \mathbf{d} \leq \mathbf{b}) \quad (10.21)$$

where columns of the matrix \mathbf{N} ($n \times p$) are the gradients of equality constraints and the columns of the matrix \mathbf{A} ($n \times m$) are the gradients of the inequality constraints. Note that since f_k is a constant that does not affect solution of the linearized subproblem, it is dropped from Eq. (10.19). Therefore, \bar{f} represents the linearized change in the original cost function. Let $\mathbf{n}^{(j)}$ and $\mathbf{a}^{(j)}$ represent the gradients of the j th equality and j th inequality constraints, respectively. Therefore, they are given as the column vectors:

$$\mathbf{n}^{(j)} = \left(\frac{\partial h_j}{\partial x_1}, \frac{\partial h_j}{\partial x_2}, \dots, \frac{\partial h_j}{\partial x_n} \right); \quad \mathbf{a}^{(j)} = \left(\frac{\partial g_j}{\partial x_1}, \frac{\partial g_j}{\partial x_2}, \dots, \frac{\partial g_j}{\partial x_n} \right) \quad (10.22)$$

The matrices \mathbf{N} and \mathbf{A} are formed using gradients of the constraints as their columns:

$$\mathbf{N} = [\mathbf{n}^{(j)}]_{(n \times p)}; \quad \mathbf{A} = [\mathbf{a}^{(j)}]_{(n \times m)} \quad (10.23)$$

Examples 10.2 and 10.3 illustrate the linearization process for nonlinear optimization problems.

EXAMPLE 10.2 Definition of Linearized Subproblem

Consider the optimization problem of Example 4.31,

$$\text{minimize } f(\mathbf{x}) = x_1^2 + x_2^2 - 3x_1x_2 \quad (\text{a})$$

subject to the constraints

$$g_1(\mathbf{x}) = \frac{1}{6}x_1^2 + \frac{1}{6}x_2^2 - 1.0 \leq 0, \quad g_2(\mathbf{x}) = -x_1 \leq 0, \quad g_3(\mathbf{x}) = -x_2 \leq 0 \quad (\text{b})$$

Linearize the cost and constraint functions about the point $\mathbf{x}^{(0)} = (1, 1)$ and write the approximate problem given by Eqs. (10.19) to (10.21).

Solution. The graphical solution for the problem is shown in Fig. 10-4. It can be seen that the optimum solution is at the point $(\sqrt{-3}, \sqrt{-3})$ with the cost function as -3 . The given point $(1, 1)$ is inside the feasible region. The gradients of cost and constraint functions are

$$\begin{aligned} \nabla f &= (2x_1 - 3x_2, 2x_2 - 3x_1), & \nabla g_1 &= \left(\frac{2}{6}x_1, \frac{2}{6}x_2 \right), \\ \nabla g_2 &= (-1, 0), & \nabla g_3 &= (0, -1) \end{aligned} \quad (\text{c})$$

Evaluating the cost and constraint functions and their gradients at the point $(1, 1)$, we get

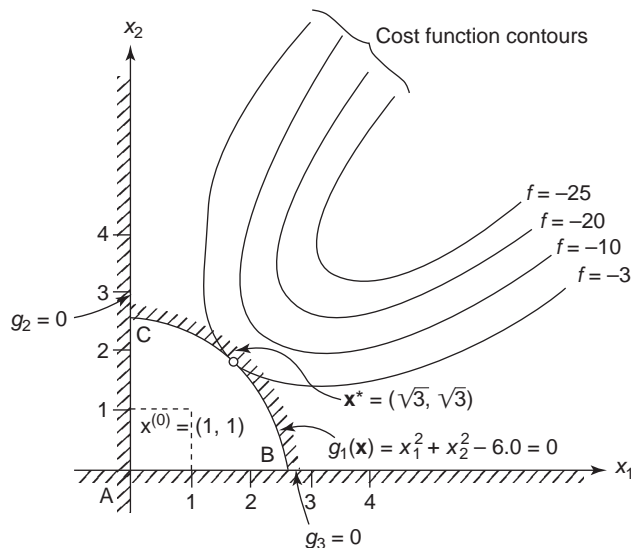


FIGURE 10-4 Graphical representation of the cost and constraints for Example 10.2.

$$\begin{aligned}
f(\mathbf{x}^{(0)}) &= -1.0, & b_1 &= -g_1(\mathbf{x}^{(0)}) = \frac{2}{3}, & b_2 &= -g_2(\mathbf{x}^{(0)}) = 1, \\
b_3 &= -g_3(\mathbf{x}^{(0)}) = 1 & \nabla f(\mathbf{x}^{(0)}) &= \mathbf{c}^{(0)} = (-1, -1), \\
\nabla g_1(\mathbf{x}^{(0)}) &= \left(\frac{1}{3}, \frac{1}{3}\right)
\end{aligned} \tag{d}$$

Note that the given design point (1, 1) is in the feasible set since all the constraints are satisfied. The matrix \mathbf{A} and vector \mathbf{b} of Eq. (10.21) are defined as

$$\mathbf{A} = \begin{bmatrix} \frac{1}{3} & -1 & 0 \\ \frac{1}{3} & 0 & -1 \\ -\frac{1}{3} & 0 & -1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \frac{2}{3} \\ 1 \\ 1 \end{bmatrix} \tag{e}$$

Now the linearized subproblem of Eqs. (10.19) to (10.21) can be written as, minimize

$$\bar{f} = [-1 \ -1] \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \tag{f}$$

subject to

$$\begin{bmatrix} \frac{1}{3} & \frac{1}{3} \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \leq \begin{bmatrix} \frac{2}{3} \\ 1 \\ 1 \end{bmatrix} \tag{g}$$

Or, in the expanded notation, we get minimize $\bar{f} = -d_1 - d_2$ subject to

$$\frac{1}{3}d_1 + \frac{1}{3}d_2 \leq \frac{2}{3}, \quad -d_1 \leq 1, \quad -d_2 \leq 1 \tag{h}$$

The last two constraints in the subproblem ensure nonnegativity of the design variables required in the problem definition. Note that unless we enforce limits on the design changes d_i , the subproblem may be unbounded.

Note also that the linearized subproblem is in terms of the design changes d_1 and d_2 . We may also write the subproblem in terms of the original variables x_1 and x_2 . To do this we replace \mathbf{d} with $\mathbf{x} - \mathbf{x}^{(0)}$ in all the foregoing expressions or in the linear Taylor's expansion and obtain:

$$\bar{f}(x_1, x_2) = f(\mathbf{x}^{(0)}) + \nabla f \cdot (\mathbf{x} - \mathbf{x}^{(0)}) = -1 + [-1 \ -1] \begin{bmatrix} (x_1 - 1) \\ (x_2 - 1) \end{bmatrix} = -x_1 - x_2 + 1 \tag{i}$$

$$\bar{g}_1(x_1, x_2) = g_1(\mathbf{x}^{(0)}) + \nabla g_1 \cdot (\mathbf{x} - \mathbf{x}^{(0)}) = -\frac{2}{3} + \left[\frac{1}{3} \ \frac{1}{3}\right] \begin{bmatrix} (x_1 - 1) \\ (x_2 - 1) \end{bmatrix} = \frac{1}{3}(x_1 + x_2 - 4) \leq 0 \tag{j}$$

$$\bar{g}_2 = -x_1 \leq 0; \quad \bar{g}_3 = -x_2 \leq 0 \tag{k}$$

In the foregoing expressions, “overbar” for a function indicates linearized approximation. The feasible regions for the linearized problem at the point (1, 1) and the original problem are shown in Fig. 10-5. Since the linearized cost function is parallel to the linearized first constraint \bar{g}_1 , the optimum solution for the linearized problem is any point on the line D–E in Fig. 10-5.

It is important to note that the linear approximations for the functions of the problem change from point to point. Therefore, the feasible region for the linearized subproblem will change with the point at which the linearization is performed.

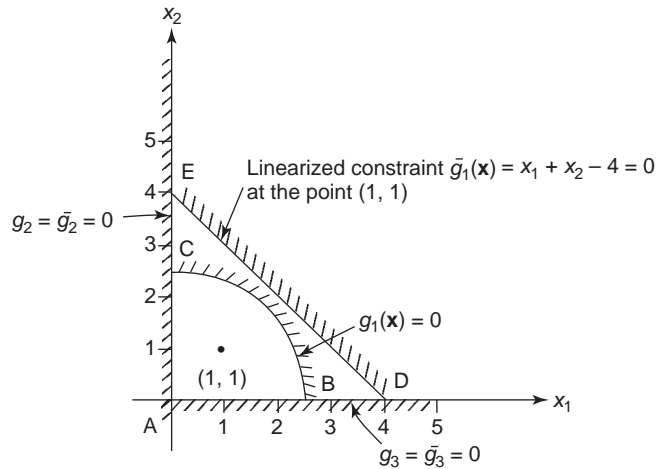


FIGURE 10-5 Graphical representation of the linearized feasible region for Example 10.2.

EXAMPLE 10.3 Linearization of Rectangular Beam Design Problem

Linearize the rectangular beam design problem formulated in Section 3.8 at the point (50, 200) mm.

Solution. The problem, after normalization, is defined as follows: Find width b and depth d to minimize $f(b, d) = bd$ subject to

$$g_1 = \frac{(2.40\text{E} + 07)}{bd^2} - 1.0 \leq 0 \quad (\text{a})$$

$$g_2 = \frac{(1.125\text{E} + 05)}{bd} - 1.0 \leq 0 \quad (\text{b})$$

$$g_3 = \frac{d}{2b} - 1.0 \leq 0 \quad (\text{c})$$

$$g_4 = -b \leq 0; \quad g_5 = -d \leq 0 \quad (\text{d})$$

At the given point the problem functions are evaluated as

$$\begin{aligned}
 f(50, 200) &= 10,000 \\
 g_1(50, 200) &= 11.00 > 0 \text{ (violation)} \\
 g_2(50, 200) &= 10.25 > 0 \text{ (violation)} \\
 g_3(50, 200) &= 1.00 > 0 \text{ (violation)} \\
 g_4(50, 200) &= -50 < 0 \text{ (inactive)} \\
 g_5(50, 200) &= -200 < 0 \text{ (inactive)}
 \end{aligned} \tag{e}$$

In the following calculations, we shall ignore constraints g_4 and g_5 assuming that they will remain satisfied, that is, the design will remain in the first quadrant. The gradients of the functions are evaluated as

$$\begin{aligned}
 \nabla f(50, 200) &= (\bar{d}, \bar{b}) = (200, 50) \\
 \nabla g_1(50, 200) &= (2.40\text{E} + 07) \left(\frac{-1}{b^2 d^2}, \frac{-2}{bd^3} \right) = (-0.24, -0.12) \\
 \nabla g_2(50, 200) &= (1.125\text{E} + 05) \left(\frac{-1}{b^2 d}, \frac{-1}{bd^2} \right) = (-0.225, -0.05625) \\
 \nabla g_3(50, 200) &= \frac{1}{2} \left(\frac{-1}{b^2} d, \frac{1}{b} \right) = (-0.04, 0.01)
 \end{aligned} \tag{f}$$

Using the function values and their gradients, the linear Taylor's expansions give the linearized subproblem at the point (50, 200) in terms of the original variables as

$$\begin{aligned}
 \bar{f}(b, d) &= 200b + 50d - 10,000 \\
 \bar{g}_1(b, d) &= -0.24b - 0.12d + 47.00 \leq 0 \\
 \bar{g}_2(b, d) &= -0.225b - 0.05625d + 32.75 \leq 0 \\
 \bar{g}_3(b, d) &= -0.04b + 0.01d + 1.00 \leq 0
 \end{aligned} \tag{g}$$

The linearized constraint functions are plotted in Fig. 10-6 and their feasible region is identified. The feasible region for the original constraints is also identified. It can be observed that the two regions are quite different. Since the linearized cost function is parallel to constraint \bar{g}_2 , the optimum solution lies on the line I-J. If point I is selected as the solution for the linearized subproblem, then the new point is given as

$$b = 95.28 \text{ mm}, \quad d = 201.10 \text{ mm}, \quad \bar{f} = 19,111 \text{ mm}^2 \tag{h}$$

For any point on line I-J all the original constraints are still violated. Apparently, for nonlinear constraints, iterations may be needed to correct constraint violations and reach the feasible set.

One interesting observation concerns the third constraint; the original constraint $d - 2b \leq 0$ is normalized as $d/2b - 1 \leq 0$. The normalization does not change the constraint boundary; thus the graphical representation for the problem remains the same, as may be verified in Fig. 10-6. However, the normalization changes the form of the

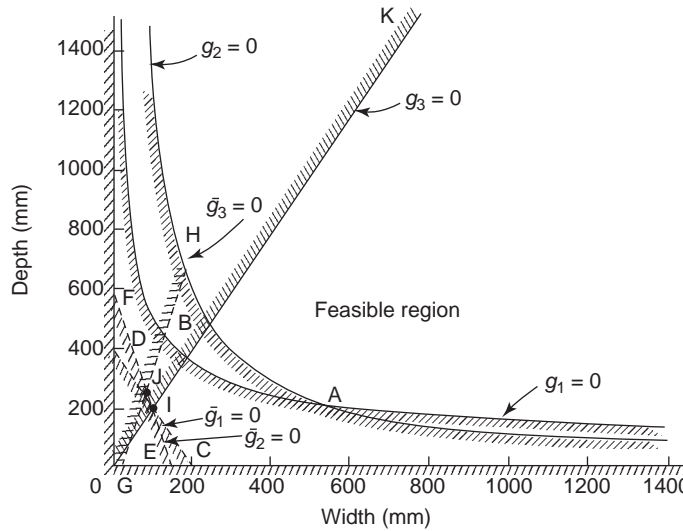


FIGURE 10-6 Feasible region for the original and the linearized constraints of the rectangular beam design problem of Example 10.3.

constraint function that affects its linearization. If the constraint is not normalized, its linearization will give the same functional form as the original constraint for all design points, i.e., $d - 2b \leq 0$. This is shown as line 0–K in Fig. 10-6. The linearized form of the normalized constraint changes; it gives the line G–H for the point (50, 200). This is quite different from the original constraint. The iterative process with and without the normalized constraint can lead to different paths to the optimum point. In conclusion, we must be careful while normalizing the constraints so as not to change the functional form for the constraints as far as possible.

10.3 Sequential Linear Programming Algorithm

Note that all the functions in Eqs. (10.19) to (10.21) are linear in the variables d_i . Therefore, linear programming methods can be used to solve for d_i . Such procedures where linear programming is used to compute design change are called *sequential linear programming* methods or in short SLP. In this section, we shall briefly describe such a procedure and discuss its advantages and drawbacks. The idea of move limits and their needs are explained and illustrated.

10.3.1 The Basic Idea—Move Limits

To solve the LP by the standard Simplex method, the right side parameters e_i and b_j in Eqs. (10.13) and (10.14) must be nonnegative. If any b_j is negative, we must multiply the corresponding constraint by -1 to make the right side nonnegative. This will change the sense of the inequality in Eq. (10.21), i.e., it will become a “ \geq type” constraint.

It must be noted that the problem defined in Eqs. (10.19) to (10.21) may not have a bounded solution, or the changes in design may become too large, thus invalidating the linear approximations. Therefore, limits must be imposed on changes in design. Such constraints are usually called *move limits*, expressed as

$$-\Delta_{il}^{(k)} \leq d_i \leq \Delta_{iu}^{(k)} \quad i = 1 \text{ to } n \quad (10.24)$$

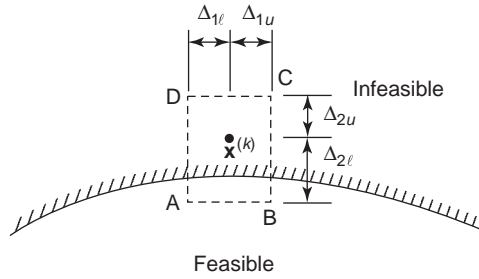


FIGURE 10-7 Linear move limits on design changes.

where $\Delta_{il}^{(k)}$ and $\Delta_{iu}^{(k)}$ are the maximum allowed decrease and increase in the i th design variable, respectively, at the k th iteration. The problem is still linear in terms of d_i so LP methods can be used to solve it. Note that the iteration counter k is used to specify $\Delta_{il}^{(k)}$ and $\Delta_{iu}^{(k)}$. That is, the move limits may change every iteration. Figure 10-7 shows the effect of imposing the move limits on changes in the design $\mathbf{x}^{(k)}$; the new design estimate is required to stay in the rectangular area ABCD for a two-dimensional problem.

Selection of Proper Move Limits Selecting proper move limits is of critical importance because it can mean success or failure of the SLP algorithm. Their specification, however, requires some experience with the method as well as knowledge of the problem being solved. Therefore, the user should not hesitate to try different move limits if one specification leads to failure or improper design. Many times lower and upper bounds are specified on the real design variables x_i . Therefore, move limits must be selected to remain within the specified bounds. Also, since linear approximations for the functions are used, the design changes should not be very large, and the move limits should not be excessively large. Usually $\Delta_{il}^{(k)}$ and $\Delta_{iu}^{(k)}$ are selected as some fraction of the current design variable values (this may vary from 1 to 100 percent). If the resulting LP problem turns out to be infeasible, the move limits will have to be relaxed (i.e., allow larger changes in design) and the subproblem solved again. Usually, a certain amount of experience with the problem is necessary to select proper move limits and adjust them at every iteration to solve the problem successfully.

Positive/Negative Design Changes Another point must also be noted before an SLP algorithm can be stated. This concerns the sign of the variables d_i (or Δx_i), which can be positive or negative, i.e., current values of design variables can increase or decrease. To allow for such a change, we must treat the LP variables d_i as free in sign. This can be done as explained in Section 6.1. Each free variable d_i is replaced as $d_i = d_i^+ - d_i^-$ in all the expressions. The LP subproblem defined in Eqs. (10.19) to (10.21) is then transformed to the standard form to use the Simplex method.

10.3.2 An SLP Algorithm

We must define some stopping criteria before stating the algorithm:

1. All constraints must be satisfied. This can be expressed as $g_i \leq \varepsilon_i$; $i = 1$ to m and $|h_i| \leq \varepsilon_i$; $i = 1$ to p , where $\varepsilon_i > 0$ is a specified small number defining tolerance for constraint violation.
2. The changes in design should be almost zero; that is, $\|\mathbf{d}\| \leq \varepsilon_2$, where $\varepsilon_2 > 0$ is a specified small number.

The *sequential linear programming* algorithm is now stated as follows:

- Step 1.* Estimate a starting design as $\mathbf{x}^{(0)}$. Set $k = 0$. Specify two small positive numbers, ε_1 and ε_2 .
- Step 2.* Evaluate cost and constraint functions at the current design $\mathbf{x}^{(k)}$, i.e., calculate f_k , b_j ; $j = 1$ to m , and e_j ; $j = 1$ to p as defined in Eqs. (10.12) to (10.14). Also, evaluate the cost and constraint function gradients at the current design $\mathbf{x}^{(k)}$.
- Step 3.* Select the proper move limits $\Delta_{ii}^{(k)}$ and $\Delta_{ij}^{(k)}$ as some fraction of the current design. Define the LP subproblem of Eqs. (10.19) to (10.21).
- Step 4.* If needed, convert the LP subproblem to the standard Simplex form (refer to Section 6.1), and solve it for $\mathbf{d}^{(k)}$.
- Step 5.* Check for convergence. If $g_i \leq \varepsilon_1$; $i = 1$ to m ; $|h_i| \leq \varepsilon_1$; $i = 1$ to p ; and $\|\mathbf{d}^{(k)}\| \leq \varepsilon_2$ then stop. Otherwise, continue.
- Step 6.* Update the design as $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$. Set $k = k + 1$ and go to Step 2.

It is interesting to note here that the LP problem defined in Eqs. (10.19) to (10.21) can be transformed to be in the original variables by substituting $d_i = x_i - x_i^{(k)}$. This was demonstrated in Examples 10.2 and 10.3. The move limits on d_i of Eq. (10.24) can also be transformed to be in the original variables. This way the solution of the LP problem directly gives the estimate for the next design point. Examples 10.4 and 10.5 illustrate the use of sequential linear programming algorithm.

EXAMPLE 10.4 Study of Sequential Linear Programming Algorithm

Consider the problem given in Example 10.2. Define the linearized subproblem at the point (3, 3), and discuss its solution imposing, proper move limits.

Solution. To define the linearized subproblem, the following quantities are calculated at the given point (3, 3):

$$f(3, 3) = 3^2 + 3^2 - 3(3)(3)(3) = -9 \quad (\text{a})$$

$$g_1(3, 3) = \frac{1}{6}(3^2) + \frac{1}{6}(3^2) - 1 = 2 > 0 \quad (\text{violation}) \quad (\text{b})$$

$$g_2(3, 3) = -x_1 = -3 < 0 \quad (\text{inactive}) \quad (\text{c})$$

$$g_3(3, 3) = -x_2 = -3 < 0 \quad (\text{inactive}) \quad (\text{d})$$

$$\mathbf{c} = \nabla f = (2x_1 - 3x_2, 2x_2 - 3x_1) = (-3, -3) \quad (\text{e})$$

$$\nabla g_1 = \left(\frac{2}{6}x_1, \frac{2}{6}x_2 \right) = (1, 1), \quad \nabla g_2 = (-1, 0), \quad \nabla g_3 = (0, -1) \quad (\text{f})$$

The given point is in the infeasible region as the first constraint is violated. The linearized subproblem is defined according to Eqs. (10.19) to (10.21) as

$$\text{minimize } \bar{f} = [-3 \quad -3] \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \quad (\text{g})$$

subject to the linearized constraints

$$\begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \leq \begin{bmatrix} -2 \\ 3 \\ 3 \end{bmatrix} \quad (\text{h})$$

The subproblem has only two variables, so it can be solved using the graphical solution procedure, as shown in Fig. 10-8. This figure when superimposed on Fig. 10-4 represents a linearized approximation for the original problem at the point (3, 3). The feasible solution for the linearized subproblem must lie in the region ABC in Fig. 10-8. The cost function is parallel to the line B-C, thus any point on the line minimizes the function. We may choose $d_1 = -1$ and $d_2 = -1$ as the solution that satisfies all the linearized constraints (note that the linearized change in cost \bar{f} is 6). If 100 percent move limits are selected, i.e., $-3 \leq d_1 \leq 3$ and $-3 \leq d_2 \leq 3$, then the solution to the LP subproblem must lie in the region ADEF. If the move limits are set as 20 percent of the current value of design variables, then the solution must satisfy $-0.6 \leq d_1 \leq 0.6$ and $-0.6 \leq d_2 \leq 0.6$. In this case the solution must lie in the region $A_1D_1E_1F_1$. It can be seen that there is no feasible solution to the linearized subproblem because region $A_1D_1E_1F_1$ does not intersect the line B-C. 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 any solution.

If we choose $d_1 = -1$ and $d_2 = -1$, then the improved design is given as (2, 2). This is still an infeasible point, as can be seen in Fig. 10-4. Therefore, although the linearized constraint is satisfied with $d_1 = -1$ and $d_2 = -1$, the original nonlinear constraint g_1 is still violated.

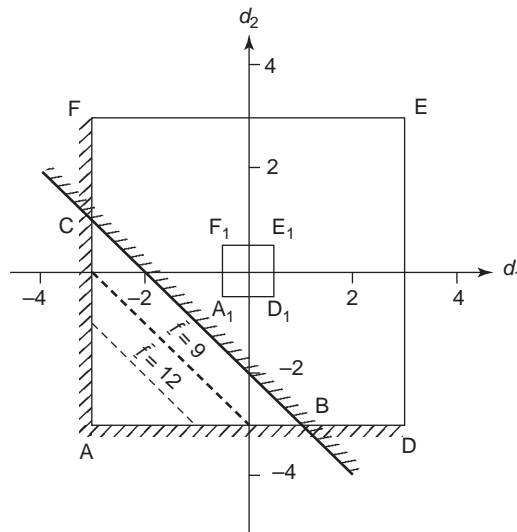


FIGURE 10-8 Graphical solution for the linearized subproblem of Example 10.4.

EXAMPLE 10.5 Use of Sequential Linear Programming

Consider the problem given in Example 10.2. Perform one iteration of the SLP algorithm. Use $\varepsilon_1 = \varepsilon_2 = 0.001$ and choose move limits such that a 15 percent design change is permissible. Let $\mathbf{x}^{(0)} = (1, 1)$ be the starting design.

Solution. The given point represents a feasible solution for the problem as may be seen in Fig. 10-4. The linearized subproblem with the appropriate move limits on design changes d_1 and d_2 at the point $\mathbf{x}^{(0)}$ is obtained in Example 10.2 as

$$\text{minimize } \bar{f} = -d_1 - d_2 \quad (\text{a})$$

subject to

$$\frac{1}{3}d_1 + \frac{1}{3}d_2 \leq \frac{2}{3} \quad (\text{b})$$

$$-(1+d_1) \leq 0, \quad -(1+d_2) \leq 0 \quad (\text{c})$$

$$-0.15 \leq d_1 \leq 0.15, \quad -0.15 \leq d_2 \leq 0.15 \quad (\text{d})$$

The graphical solution for the linearized subproblem is given in Fig. 10-9. Move limits of 15 percent define the solution region as DEFG. The optimum solution for the problem is at point F where $d_1 = 0.15$ and $d_2 = 0.15$. It is seen that much larger move limits are possible in the present case.

We shall solve the problem using the Simplex method as well. Note that in the linearized subproblem, the design changes d_1 and d_2 are free in sign. If we wish to solve the problem by the Simplex method, we must define new variables, A , B , C , and D such that $d_1 = A - B$, $d_2 = C - D$ and A , B , C , and $D \geq 0$. Therefore, substituting these

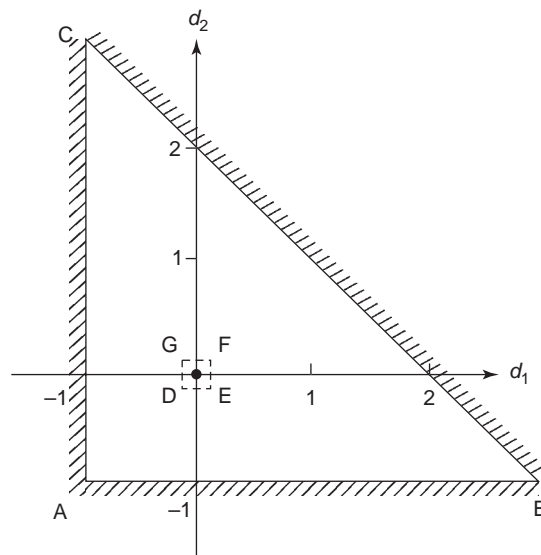


FIGURE 10-9 Graphical solution for the linearized subproblem of Example 10.5.

decompositions into the foregoing equations, we get the following problem written in standard form as:

$$\text{minimize } \bar{f} = -A + B - C + D \quad (\text{e})$$

subject to

$$\frac{1}{3}(A - B + C - D) \leq \frac{2}{3} \quad (\text{f})$$

$$-A + B \leq 1.0, \quad -C + D \leq 1.0 \quad (\text{g})$$

$$A - B \leq 0.15, \quad B - A \leq 0.15 \quad (\text{h})$$

$$C - D \leq 0.15, \quad D - C \leq 0.15 \quad (\text{i})$$

$$A, B, C, D \geq 0 \quad (\text{j})$$

The solution to the foregoing LP problem with the Simplex method is $A = 0.15$, $B = 0$, $C = 0.15$, and $D = 0$. Therefore, $d_1 = A - B = 0.15$ and $d_2 = C - D = 0.15$. This gives the updated design as $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \mathbf{d}^{(0)} = (1.15, 1.15)$. At the new design $(1.15, 1.15)$, we have, $f(\mathbf{x}^{(1)}) = -1.3225$ and $g_1(\mathbf{x}^{(1)}) = -0.5592$. Note that the cost function has decreased for the new design $\mathbf{x}^{(1)}$ without violating the constraint. This indicates that the new design is an improvement over the previous one. Since the norm of the design change, $\|\mathbf{d}\| = 0.212$ is larger than the permissible tolerance (0.001), we need to go through more iterations to satisfy the stopping criteria.

Note also that the linearized subproblem at the point $(1, 1)$ can be written in the original variables. This was done in Example 10.2 and the linearized subproblem was obtained as

$$\text{minimize } \bar{f} = -x_1 - x_2 + 1 \quad (\text{k})$$

subject to

$$\bar{g}_1 = \frac{1}{3}(x_1 + x_2 - 4) \leq 0, \quad \bar{g}_2 = -x_1 \leq 0, \quad \bar{g}_3 = -x_2 \leq 0 \quad (\text{l})$$

The 15 percent move limits can be also transformed to be in the original variables using $-\Delta_{ii} \leq x_i - x_i^{(0)} \leq \Delta_{ii}$ as

$$-0.15 \leq (x_1 - 1) \leq 0.15 \quad \text{or} \quad 0.85 \leq x_1 \leq 1.15 \quad (\text{m})$$

$$-0.15 \leq (x_2 - 1) \leq 0.15 \quad \text{or} \quad 0.85 \leq x_2 \leq 1.15 \quad (\text{n})$$

Solving the subproblem, we obtain the same solution $(1.15, 1.15)$ as before.

10.3.3 SLP Algorithm: Some Observations

The sequential linear programming algorithm is a simple and straightforward approach to solving constrained optimization problems. The algorithm can be used to solve various engineering design problems, especially problems having a large number of design variables. The following observations highlight some features and limitations of the SLP method.

1. *The method should not be used as a black box approach for engineering design problems.* The selection of move limits is a trial and error process and can be best achieved in an interactive mode. The move limits can be too restrictive resulting in no solution for the LP subproblem. Move limits that are too large can cause oscillations in the design point during iterations. Thus performance of the method depends heavily on selection of move limits.
2. *The method may not converge to the precise minimum* since no descent function is defined, and line search is not performed along the search direction to compute a step size. Thus progress toward the solution point cannot be monitored.
3. *The method can cycle between two points* if the optimum solution is not a vertex of the feasible set.
4. The method is quite simple conceptually as well as numerically. Although it may not be possible to reach the precise optimum with the method, it can be used to obtain improved designs in practice.

10.4 Quadratic Programming Subproblem

As observed in the previous section, the SLP is a simple algorithm to solve general constrained optimization problems. However, the method has some limitations, the major one being the lack of robustness. To correct the drawbacks, a method is presented in the next section where a quadratic programming (QP) subproblem is solved to determine a search direction. Then a step size is calculated by minimizing a descent function along the search direction. In this section, we shall define a QP subproblem and discuss a method for solving it.

10.4.1 Definition of QP Subproblem

To overcome some of the limitations of the SLP method, other methods have been developed to solve for design changes. Most of the methods still utilize linear approximations of Eqs. (10.19) to (10.21) for the nonlinear optimization problem. However, the linear move limits of Eq. (10.24) are abandoned in favor of a step size calculation procedure. The move limits of Eq. (10.24) play two roles in the solution process: (1) they make the linearized subproblem bounded, and (2) they give the design change without performing line search. It turns out that these two roles of the move limits of Eq. (10.24) can be achieved by defining and solving a slightly different subproblem to determine the search direction and then performing a line search for the step size to calculate the design change. The linearized subproblem can be bounded if we require minimization of the length of the search direction in addition to minimization of the linearized cost function. This can be accomplished by combining these two objectives. Since this combined objective is a quadratic function in terms of the search direction, the resulting subproblem is called a QP subproblem. The subproblem is defined as

$$\text{minimize } \bar{f} = \mathbf{c}^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \mathbf{d} \quad (10.25)$$

subject to the linearized constraints of Eqs. (10.20) and (10.21)

$$\mathbf{N}^T \mathbf{d} = \mathbf{e}; \quad \mathbf{A}^T \mathbf{d} \leq \mathbf{b} \quad (10.26)$$

The factor of $\frac{1}{2}$ with the second term in Eq. (10.25) is introduced to eliminate the factor of 2 during differentiations. Also, the square of the length of \mathbf{d} is used instead of the length

of **d**. Note that the QP subproblem is strictly convex and therefore its minimum (if one exists) is global and unique. It is also important to note that the cost function of Eq. (10.25) represents an equation of a hypersphere with center at $-\mathbf{c}$. Example 10.6 demonstrates how to define a quadratic programming subproblem at a given point.

EXAMPLE 10.6 Definition of a QP Subproblem

Consider the constrained optimization problem:

$$\text{minimize } f(\mathbf{x}) = 2x_1^2 + 15x_2^2 - 8x_1x_2 - 4x_1 \quad (\text{a})$$

subject to equality and inequality constraints as

$$h(\mathbf{x}) = x_1^2 + x_1x_2 + 1.0 = 0, \quad g(\mathbf{x}) = x_1 - \frac{1}{4}x_2^2 - 1.0 \leq 0 \quad (\text{b})$$

Linearize the cost and constraint functions about a point (1, 1) and define the QP subproblem.

Solution. Figure 10-10 shows a graphical representation for the problem. Note that the constraints are already written in the normalized form. The equality constraint is shown as $h = 0$ and boundary of the inequality constraint as $g = 0$. The feasible region for the inequality constraint is identified and several cost function contours are shown. Since the equality constraint must be satisfied the optimum point must lie on the two curves $h = 0$. Two optimum solutions are identified as

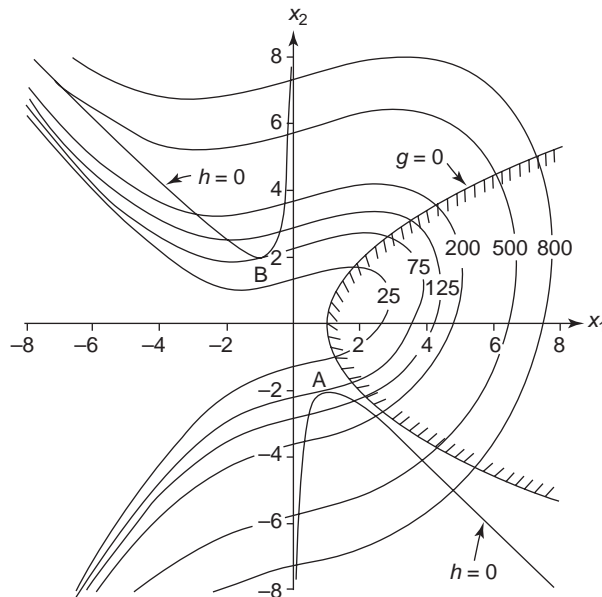


FIGURE 10-10 Graphical representation of Example 10.6.

Point A:

$$\mathbf{x}^* = (1, -2), \quad f(\mathbf{x}^*) = 74$$

Point B:

$$\mathbf{x}^* = (-1, 2), \quad f(\mathbf{x}^*) = 78$$

The gradients of the cost and constraint functions are

$$\nabla f = (6x_1^2 - 8x_2 - 4, 30x_2 - 8x_1), \quad \nabla h = (2x_1 + x_2, x_1), \quad \nabla g = (1, -x_2/2) \quad (c)$$

The cost and constraint function values and their gradients at (1, 1) are

$$\begin{aligned} f(1, 1) = 5, \quad h(1, 1) = 3 \neq 0 \text{ (violation)}, \quad g(1, 1) = -0.25 < 0 \text{ (inactive)} \\ \mathbf{c} = \nabla f = (-6, 22), \quad \nabla h = (3, 1), \quad \nabla g = (1, -0.5) \end{aligned} \quad (d)$$

Using move limits of 50 percent, the linear programming subproblem of Eqs. (10.19) to (10.21) is defined as:

$$\text{minimize } \bar{f} = -6d_1 + 22d_2 \quad (e)$$

subject to

$$3d_1 + d_2 = -3, \quad d_1 - 0.5d_2 \leq 0.25, \quad -0.5 \leq d_1 \leq 0.5, \quad -0.5 \leq d_2 \leq 0.5 \quad (f)$$

The QP subproblem of Eqs. (10.25) and (10.26) is defined as:

$$\text{minimize } \bar{f} = (-6d_1 + 22d_2) + \frac{1}{2}(d_1^2 + d_2^2) \quad (g)$$

subject to

$$3d_1 + d_2 = -3, \quad d_1 - 0.5d_2 \leq 0.25 \quad (h)$$

To compare the solutions, the preceding LP and QP subproblems are plotted in Figs. 10-11 and 10-12, respectively. In these figures, the solution must satisfy the linearized equality constraint, so it must lie on the line C–D. The feasible region for the linearized inequality constraint is also shown. Therefore, the solution for the subproblem must lie on the line G–C. It can be seen in Fig. 10-11 that with 50 percent move limits, the linearized subproblem is infeasible. The move limits require the changes to lie in the square HIJK, which does not intersect the line G–C. If we relax the move limits to 100 percent, then point L gives the optimum solution: $d_1 = -\frac{2}{3}$, $d_2 = -1.0$, $\bar{f} = -18$. Thus, we again see that the design change with the linearized subproblem is affected by the move limits.

With the QP subproblem, the constraint set remains the same but there is no need for the move limits as seen in Fig. 10-12. The cost function is quadratic in variables. The optimum solution is at point G: $d_1 = -0.5$, $d_2 = -1.5$, $\bar{f} = -28.75$. Note that the direction determined by the QP subproblem is unique, but it depends on the move limits with the LP subproblem. The two directions determined by LP and QP subproblems are in general different.

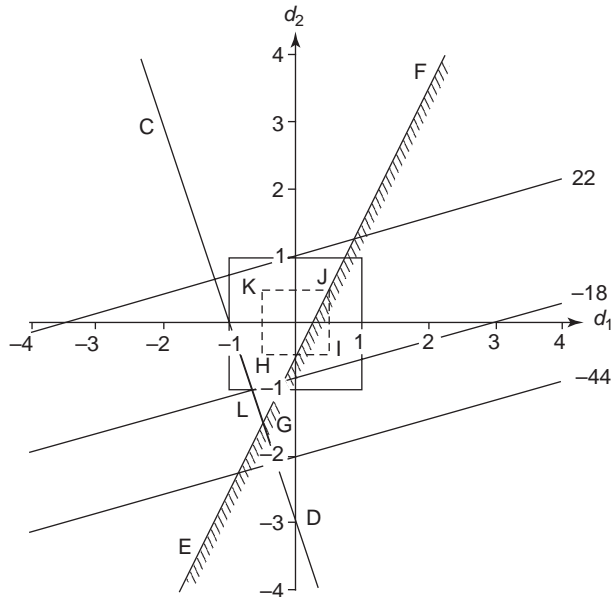


FIGURE 10-11 Solution of the linearized subproblem for Example 10.6 at the point (1,1).

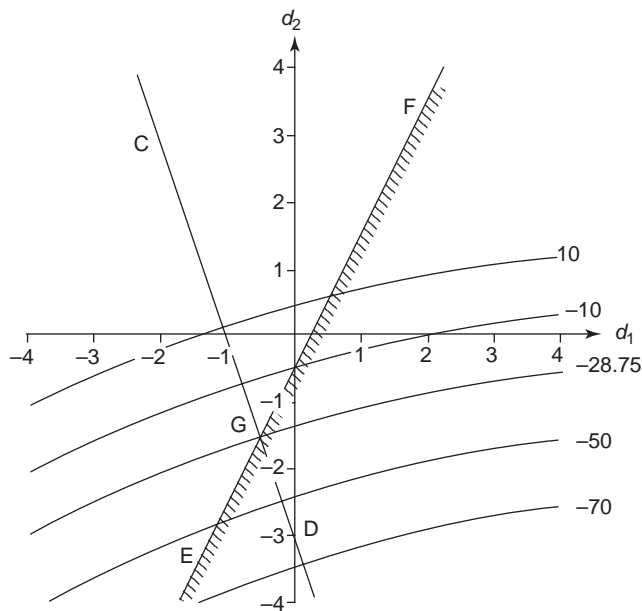


FIGURE 10-12 Solution of the quadratic programming subproblem for Example 10.6 at the point (1,1).

10.4.2 Solution of QP Subproblem

QP problems are encountered in many real-world applications. In addition, many general nonlinear programming algorithms require solution of a quadratic programming subproblem at each design cycle. Therefore it is extremely important to solve a QP subproblem efficiently

so that large-scale optimization problems can be treated. Thus, it is not surprising that substantial research effort has been expended in developing and evaluating many algorithms for solving QP problems (Gill *et al.*, 1981; Luenberger, 1984). Also, many good programs have been developed to solve such problems. In the next chapter, we shall describe a method for solving general QP problems that is a simple extension of the Simplex method of linear programming. If the problem is simple, we can solve it using the KKT conditions of optimality given in Theorem 4.6. To aid the KKT solution process, we can use a graphical representation of the problem to identify the possible solution case and solve that case only. We present such a procedure in Example 10.7.

EXAMPLE 10.7 Solution of QP Subproblem

Consider the problem of Example 10.2 linearized as: minimize $\bar{f} = -d_1 - d_2$ subject to $\frac{1}{3}d_1 + \frac{1}{3}d_2 \leq \frac{2}{3}$, $-d_1 \leq 1$, $-d_2 \leq 1$. Define the quadratic programming subproblem and solve it.

Solution. The linearized cost function is modified to a quadratic function as follows:

$$\bar{f} = (-d_1 - d_2) + 0.5(d_1^2 + d_2^2) \quad (\text{a})$$

The cost function corresponds to an equation of a circle with center at $(-c_1, -c_2)$ where c_i are components of the gradient of the cost function; i.e., at $(1, 1)$. The graphical solution for the problem is shown in Fig. 10-13 where triangle ABC represents the feasible set. Cost function contours are circles of different radii. The optimum solution is at point D where $d_1 = 1$ and $d_2 = 1$. Note that the QP subproblem is strictly convex and thus, has a unique solution. A numerical method must generally be used

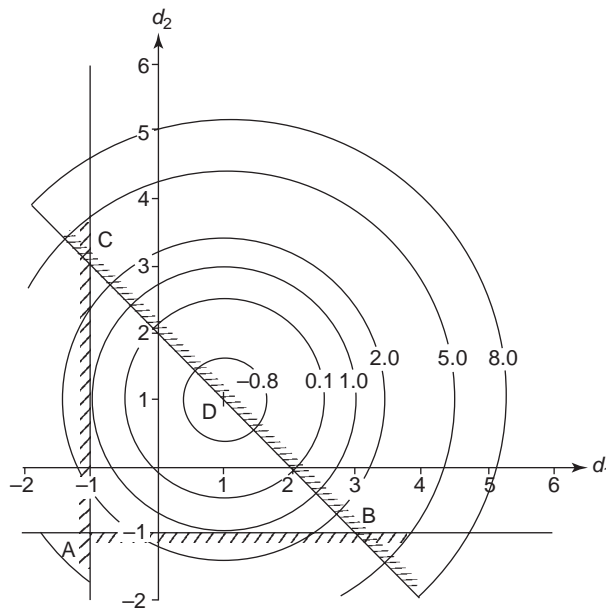


FIGURE 10-13 Solution of quadratic programming subproblem for Example 10.7 at the point $(1, 1)$.

to solve the subproblem. However, since the present problem is quite simple, it can be solved by writing the KKT necessary conditions of Theorem 4.6 as follows:

$$L = (-d_1 - d_2) + 0.5(d_1^2 + d_2^2) + u_1 \left(\frac{1}{3}(d_1 + d_2 - 2) + s_1^2 \right) + u_2(-d_1 - 1 + s_2^2) + u_3(-d_2 - 1 + s_3^2) \quad (\text{b})$$

$$\frac{\partial L}{\partial d_1} = -1 + d_1 + \frac{1}{3}u_1 - u_2 = 0, \quad \frac{\partial L}{\partial d_2} = -1 + d_2 + \frac{1}{3}u_1 - u_3 = 0 \quad (\text{c})$$

$$\frac{1}{3}(d_1 + d_2 - 2) + s_1^2 = 0 \quad (\text{d})$$

$$(-d_1 - 1) + s_2^2 = 0; \quad (-d_2 - 1) + s_3^2 = 0 \quad (\text{e})$$

$$u_i s_i = 0, \quad u_i \geq 0, \quad s_i^2 \geq 0, \quad i = 1, 2, 3 \quad (\text{f})$$

where u_1 , u_2 , and u_3 are the Lagrange multipliers for the three constraints and s_1^2 , s_2^2 , and s_3^2 are the corresponding slack variables. Note that the switching conditions $u_i s_i = 0$ give eight solution cases. However, only one case can give the optimum solution. The graphical solution shows that only the first inequality is active at the optimum, giving the case as: $s_1 = 0$, $u_2 = 0$, $u_3 = 0$. Solving this case, we get the direction vector $\mathbf{d} = (1, 1)$ with $\bar{f} = -1$ and $\mathbf{u} = (0, 0, 0)$, which is the same as the graphical solution.

10.5 Constrained Steepest Descent Method

As noted at the beginning of this chapter, numerous methods have been proposed and evaluated for constrained optimization problems since 1960. Some methods have good performance for equality constrained problems only, whereas others have good performance for inequality constrained problems only. An overview of some of these methods is presented later in Chapter 11. In this section, we focus only on a general method, called the constrained steepest descent method, that can treat equality as well as inequality constraints in its computational steps. It also requires inclusion of only a few of the critical constraints in the calculation of the search direction at each iteration; that is, the QP subproblem of Eqs. (10.25) and (10.26) may be defined using only the active and violated constraints. This can lead to efficiency of calculations for larger scale engineering design problems, as explained in Chapter 11. The method has been proved to be convergent to a local minimum point starting from any point. *This is considered as a model algorithm that illustrates how most optimization algorithms work.* In addition, it can be extended for more efficient calculations, as explained in Chapter 11. Here, we explain the method and illustrate its calculations with simple numerical examples. A descent function and a step size determination procedure for the method are described. A step-by-step procedure is given to show the kind of calculations needed to implement the method for numerical calculations. It is important to understand these steps and calculations to effectively use optimization software and diagnose errors when something goes wrong with an application.

Note that when there are either no constraints or no active ones, minimization of the quadratic function of Eq. (10.25) gives $\mathbf{d} = -\mathbf{c}$ (using the necessary condition, $\partial \bar{f} / \partial \mathbf{d} = \mathbf{0}$). This is just the steepest descent direction of Section 8.3 for the unconstrained problems. When

there are constraints, their effect must be included in calculating the search direction. The search direction must satisfy all the linearized constraints. Since the search direction is a modification of the steepest descent direction to satisfy constraints, it is called the *constrained steepest descent direction*. The steps of the resulting *constrained steepest descent algorithm* (CSD) will be clear once we define a suitable *descent function and a related line search procedure to calculate the step size* along the search direction. It is important to note that the CSD method presented in this section is the most introductory and simple interpretation of more powerful *sequential quadratic programming* (SQP) methods. All features of the algorithms are not discussed here to keep the presentation of the key ideas simple and straightforward. It is noted, however, that the methods work equally well when initiated from feasible or infeasible points.

10.5.1 Descent Function

Recall that in unconstrained optimization methods the cost function is used as the descent function to monitor progress of algorithms toward the optimum point. For constrained problems, the descent function is usually constructed by adding a *penalty* for constraint violations to the current value of the cost function. Based on this idea, many descent functions can be formulated. In this section, we shall describe one of them and show its use.

One of the properties of a descent function is that its value at the optimum point must be the same as that for the cost function. Also, it should be such that a unit step size is admissible in the neighborhood of the optimum point. We shall introduce *Pshenichny's descent function* (also called the exact penalty function) because of its simplicity and success in solving a large number of engineering design problems (Pshenichny and Danilin, 1982; Belegundu and Arora, 1984a,b). Other descent functions shall be discussed in Chapter 11. Pshenichny's descent function Φ at any point \mathbf{x} is defined as

$$\Phi(\mathbf{x}) = f(\mathbf{x}) + RV(\mathbf{x}) \quad (10.27)$$

where $R > 0$ is a positive number called the *penalty parameter* (initially specified by the user), $V(\mathbf{x}) \geq 0$ is either the *maximum constraint violation* among all the constraints or zero, and $f(\mathbf{x})$ is the cost function value at \mathbf{x} . As an example, the descent function at the point $\mathbf{x}^{(k)}$ during the k th iteration is calculated as

$$\Phi_k = f_k + RV_k \quad (10.28)$$

where Φ_k and V_k are the values of $\Phi(\mathbf{x})$ and $V(\mathbf{x})$ at $\mathbf{x}^{(k)}$ as

$$\Phi_k = \Phi(\mathbf{x}^{(k)}); \quad V_k = V(\mathbf{x}^{(k)}) \quad (10.29)$$

and R is the most current value of the penalty parameter. As explained later with examples, the penalty parameter may change during the iterative process. Actually, it must be ensured that it is greater than or equal to the sum of all the Lagrange multipliers of the QP subproblem at the point $\mathbf{x}^{(k)}$. This is a necessary condition given as

$$R \geq r_k \quad (10.30)$$

where r_k is the sum of all the Lagrange multipliers at the k th iteration:

$$r_k = \sum_{i=1}^p |v_i^{(k)}| + \sum_{i=1}^m u_i^{(k)} \quad (10.31)$$

Since the Lagrange multiplier $v_i^{(k)}$ for an equality constraint is free in sign, its absolute value is used in Eq. (10.31). $u_i^{(k)}$ is the multiplier for the i th inequality constraint.

The parameter $V_k \geq 0$ related to the maximum constraint violation at the k th iteration is determined using the calculated values of the constraint functions at the design point $\mathbf{x}^{(k)}$ as

$$V_k = \max\{0; |h_1|, |h_2|, \dots, |h_p|; g_1, g_2, \dots, g_m\} \quad (10.32)$$

Since the equality constraint is violated if it is different from zero, the absolute value is used with each h_i in Eq. (10.32). Note that V_k is always nonnegative, i.e., $V_k \geq 0$. If all constraints are satisfied at $\mathbf{x}^{(k)}$, then $V_k = 0$. Example 10.8 illustrates calculations for the descent function.

EXAMPLE 10.8 Calculation of Descent Function

A design problem is formulated as follows:

$$\text{minimize } f(\mathbf{x}) = x_1^2 + 320x_1x_2 \quad (a)$$

subject to four inequalities

$$\frac{x_1}{60x_2} - 1 \leq 0, \quad 1 - \frac{x_1(x_1 - x_2)}{3600} \leq 0, \quad -x_1 \leq 0, \quad -x_2 \leq 0 \quad (b)$$

Taking the penalty parameter R as 10,000, calculate the value of the descent function at the point $\mathbf{x}^{(0)} = (40, 0.5)$.

Solution. The cost and constraint functions at the given point $\mathbf{x}^{(0)} = (40, 0.5)$ are evaluated as

$$f_0 = f(40, 0.5) = (40)^2 + 320(40)(0.5) = 8000 \quad (c)$$

$$g_1 = \frac{40}{60(0.5)} - 1 = 0.333 \text{ (violation)} \quad (d)$$

$$g_2 = 1 - \frac{40(40 - 0.5)}{3600} = 0.5611 \text{ (violation)} \quad (e)$$

$$g_3 = -40 < 0 \text{ (inactive)} \quad (f)$$

$$g_4 = -0.5 < 0 \text{ (inactive)} \quad (g)$$

Thus, the maximum constraint violation is determined using Eq. (10.32) as

$$V_0 = \max\{0; 0.333, 0.5611, -40, -0.5\} = 0.5611 \quad (h)$$

Using Eq. (10.28), the descent function is calculated as

$$\Phi_0 = f_0 + RV_0 = 8000 + (10,000)(0.5611) = 13,611 \quad (i)$$

10.5.2 Step Size Determination

Before the constrained steepest descent algorithm can be stated, a step size determination procedure is needed. The step size determination problem is to calculate α_k for use in Eq. (10.4) that minimizes the descent function Φ of Eq. (10.27). In most practical implementations of the algorithm, an *inaccurate line search* that has worked fairly well is used to determine the step size. We shall describe that procedure and illustrate its use with examples in Chapter 11. In this section we assume that a step size along the search direction can be calculated using the golden section method described in Chapter 8. However, it is realized that the method can be inefficient; therefore, inaccurate line search should be preferred in most constrained optimization methods.

In performing the line search for minimum of the descent function Φ , we need a notation to represent the trial design points, and values of the descent, cost, and constraint functions. The following notation shall be used at iteration k :

- α_j : j th trial step size
- $x_i^{(k,j)}$: i th design variable value at the j th trial step size
- $f_{k,j}$: cost function value at the j th trial point
- $\Phi_{k,j}$: descent function value at the j th trial point
- $V_{k,j}$: maximum constraint function value at the j th trial point
- R_k : penalty parameter value that is kept fixed during line search as long as the necessary condition of Eq. (10.30) is satisfied.

Example 10.9 illustrates calculations for the descent function during golden section search.

EXAMPLE 10.9 Calculation of Descent Function for Golden Section Search

For the design problem defined in Example 10.8, the QP subproblem has been defined and solved at the starting point $\mathbf{x}^{(0)} = (40, 0.5)$. The search direction is determined as $\mathbf{d}^{(0)} = (26.60, 0.45)$ and the Lagrange multipliers for the constraints are determined as $\mathbf{u} = (4880, 19\,400, 0, 0)$. Let the initial value of the penalty parameter be given as $R_0 = 1$. Calculate the descent function value at the two points during initial bracketing of the step size in the golden section search using $\delta = 0.1$. Compare the descent function values.

Solution. Since we are evaluating the step size at the starting point, $k = 0$, and j will be taken as 0, 1, and 2. Using the calculations given in Example 10.8 at the starting point, we get

$$f_{0,0} = 8000, \quad V_{0,0} = 0.5611 \quad (\text{a})$$

To check the necessary condition of Eq. (10.30) for the penalty parameter, we need to evaluate r_0 using Eq. (10.31) as follows:

$$r_0 = \sum_{i=1}^m u_i^{(0)} = 4880 + 19,400 + 0 + 0 = 24,280 \quad (\text{b})$$

The necessary condition of Eq. (10.30) is satisfied if we select the penalty parameter R as $R = \max(R_0, r_0)$:

$$R = \max(1, 24\,280) = 24,280 \quad (\text{c})$$

Thus, the descent function value at the starting point is given as

$$\Phi_{0,0} = f_{0,0} + RV_{0,0} = 8000 + (24,280)(0.5611) = 21,624 \quad (d)$$

Now let us calculate the descent function at the first trial step size $\delta = 0.1$, i.e., $\alpha_1 = 0.1$. Updating the current design point in the search direction, we get

$$\mathbf{x}^{(0,1)} = \begin{bmatrix} 40 \\ 0.5 \end{bmatrix} + (0.1) \begin{bmatrix} 25.6 \\ 0.45 \end{bmatrix} = \begin{bmatrix} 42.56 \\ 0.545 \end{bmatrix} \quad (e)$$

Various functions for the problem are calculated at $\mathbf{x}^{(0,1)}$ as

$$f = 9233.8, \quad g_1 = 0.3015, \quad g_2 = 0.5033, \quad g_3 = -42.56, \quad g_4 = -0.545 \quad (f)$$

The constraint violation parameter is given as

$$V_{0,1} = \max\{0; 0.3015, 0.5033, -42.56, -0.545\} = 0.5033 \quad (g)$$

Thus, the descent function at the trial step size of $\alpha_1 = 0.1$ is given as (note that the value of the penalty parameter R is not changed during step size calculation)

$$\Phi_{0,1} = f_{0,1} + RV_{0,1} = 9233.8 + (24,280)(0.5033) = 21,454 \quad (h)$$

Since $\Phi_{0,1} < \Phi_{0,0}$ ($21,454 < 21,624$), we need to continue the initial bracketing process in the golden section search procedure. Following that procedure, the next trial step size is given as $\alpha_2 = \delta + 1.618\delta = 0.1 + 1.618(0.1) = 0.2618$. The trial design point is obtained as

$$\mathbf{x}^{(0,1)} = \begin{bmatrix} 40 \\ 0.5 \end{bmatrix} + (0.2618) \begin{bmatrix} 25.6 \\ 0.45 \end{bmatrix} = \begin{bmatrix} 46.70 \\ 0.618 \end{bmatrix} \quad (i)$$

Following the foregoing procedure, various quantities are calculated as

$$f = 11,416.3, \quad g_1 = 0.2594, \quad g_2 = 0.4022, \quad g_3 = -46.70, \quad g_4 = -0.618 \quad (j)$$

$$V_{0,1} = \max\{0; 0.2594, 0.4022, -46.70, -0.618\} = 0.4022 \quad (k)$$

$$\Phi_{0,2} = f_{0,2} + RV_{0,2} = 11,416.3 + (24,280)(0.4022) = 21,182 \quad (l)$$

Since $\Phi_{0,2} < \Phi_{0,1}$ ($21,182 < 21,454$), the minimum for the descent function has not been surpassed yet. Therefore we need to continue the initial bracketing process. The next trial step size is given as

$$\alpha_3 = \delta + 1.618\delta + 1.618^2\delta = 0.1 + 1.618(0.1) + 2.618(0.1) = 0.5236 \quad (m)$$

Following the foregoing procedure, $\Phi_{0,3}$ can be calculated and compared with $\Phi_{0,2}$.

Note that the value of the penalty parameter R is calculated at the beginning of the line search and then kept fixed during all subsequent calculations for step size determination.

10.5.3 CSD Algorithm

We are now ready to state the constrained steepest descent (CSD) algorithm in a step-by-step form. It has been proved that the solution point of the sequence $\{\mathbf{x}^{(k)}\}$ generated by the algorithm is a KKT point for the general constrained optimization problem (Pshenichny and Danilin, 1982). The stopping criterion for the algorithm is that $\|\mathbf{d}\| \leq \varepsilon$ for a feasible point. Here ε is a small positive number and \mathbf{d} is the search direction that is obtained as a solution of the QP subproblem. The CSD method is now summarized in the form of a *computational algorithm*.

- Step 1.* Set $k = 0$. Estimate initial values for design variables as $\mathbf{x}^{(0)}$. Select an appropriate initial value for the penalty parameter R_0 , and two small numbers ε_1 and ε_2 that define the permissible constraint violation and convergence parameter values, respectively. $R_0 = 1$ is a reasonable selection.
- Step 2.* At $\mathbf{x}^{(k)}$ compute the cost and constraint functions and their gradients. Calculate the maximum constraint violation V_k as defined in Eq. (10.32).
- Step 3.* Using the cost and constraint function values and their gradients, define the QP subproblem given in Eqs. (10.25) and (10.26). Solve the QP subproblem to obtain the search direction $\mathbf{d}^{(k)}$ and Lagrange multipliers vectors $\mathbf{v}^{(k)}$ and $\mathbf{u}^{(k)}$.
- Step 4.* Check for the following stopping criteria $\|\mathbf{d}^{(k)}\| \leq \varepsilon_2$ and the maximum constraint violation $V_k \leq \varepsilon_1$. If these criteria are satisfied then stop. Otherwise continue.
- Step 5.* To check the necessary condition of Eq. (10.30) for the penalty parameter R , calculate the sum r_k of the Lagrange multipliers defined in Eq. (10.31). Set $R = \max\{R_k, r_k\}$. This will always satisfy the necessary condition of Eq. (10.30).
- Step 6.* Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$, where $\alpha = \alpha_k$ is a proper step size. As for the unconstrained problems, the step size can be obtained by minimizing the descent function of Eq. (10.27) along the search direction $\mathbf{d}^{(k)}$. Any of the procedures, such as the golden section search, can be used to determine a step size.
- Step 7.* Save the current penalty parameter as $R_{k+1} = R$. Update the iteration counter as $k = k + 1$, and go to Step 2.

The CSD algorithm along with the foregoing step size determination procedure is convergent provided second derivatives of all the functions are piece-wise continuous (this is the so-called Lipschitz condition) and the set of design points $\mathbf{x}^{(k)}$ are bounded as follows:

$$\Phi(\mathbf{x}^{(k)}) \leq \Phi(\mathbf{x}^{(0)}); \quad k = 1, 2, 3, \dots$$

10.5.4 CSD Algorithm: Some Observations

1. The CSD algorithm is a *first-order* method that can treat equality and inequality constraints. The algorithm converges to a local minimum point starting from an arbitrary point.
2. The *potential constraint strategy* discussed in the next chapter is not introduced in the algorithm for the sake of simplicity of presentation. This strategy is essential for engineering applications and can be easily incorporated into the algorithm (Belegundu and Arora, 1984).
3. The golden section search can be inefficient and is generally not recommended for engineering applications. The inaccurate line search described in Chapter 11 works quite well and is recommended.
4. The rate of convergence of the CSD algorithm can be improved by including second-order information in the QP subproblem. This is discussed in Chapter 11.

- The starting point can affect performance of the algorithm. For example, at some points the QP subproblem may not have any solution. This need not mean that the original problem is infeasible. The original problem may be highly nonlinear, so that the linearized constraints may be inconsistent giving infeasible subproblem. This situation can be handled by either temporarily deleting the inconsistent constraints or starting from another point. For more discussion on the implementation of the algorithm, Tseng and Arora (1988) may be consulted.

10.6 Engineering Design Optimization Using Excel Solver

Project/Problem Statement Welded plate girders are used in many practical applications, such as overhead cranes, and highway and railway bridges. As an example of formulation of a practical design problem and the optimization solution process, we shall present design of a welded plate girder for a highway bridge to minimize its cost. Other applications of plate girders can be formulated and solved in a similar way. It has been determined that the life-cycle cost of the girder is related to its total mass. Since mass is proportional to the material volume, the objective of this project is to design a minimum volume girder and at the same time satisfy requirements of the AASHTO Specifications (American Association of State Highway and Transportation Officials) (Arora *et al.*, 1997). The dead load for the girder consists of weight of the pavement and self weight of the girder. The live load consists of equivalent uniform load and concentrated loads based on HS-20(MS18) truck loading condition. The cross-section of the girder is shown in Fig. 10-14.

In this section, we present a formulation for the problem using the procedure described in Chapter 2. Preparation of the Excel worksheet to solve the problem is explained and the problem is solved using Solver.

Data and Information Collection Material and loading data and other parameters for the plate girder are specified as follows:

Span:	$L = 25 \text{ m}$
Modulus of elasticity:	$E = 210 \text{ GPa}$

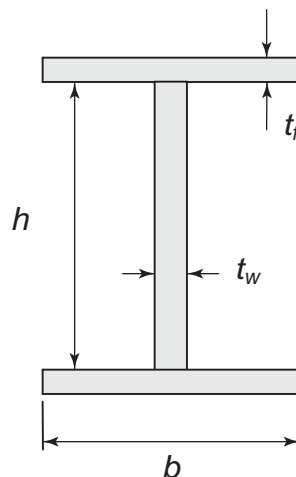


FIGURE 10-14 Cross-section of plate girder.

Yield stress:	$\sigma_y = 262 \text{ MPa}$
Allowable bending stress:	$\sigma_a = 0.55\sigma_y = 144.1 \text{ MPa}$
Allowable shear stress:	$\tau_a = 0.33\sigma_y = 86.46 \text{ MPa}$
Allowable fatigue stress:	$\sigma_f = 255 \text{ MPa}$
Allowable deflection:	$D_a = \frac{L}{800}, \text{ m}$
Concentrated load for moment:	$P_m = 104 \text{ kN}$
Concentrated load for shear:	$P_s = 155 \text{ kN}$
Live load impact factor:	$LLIF = 1 + \frac{50}{(L+125)}$

Note that the live load impact factor depends on the span length L . For $L = 25 \text{ m}$, this factor is calculated as 1.33, and it is assumed that the loads P_m and P_s already incorporate this factor. The dependent variables for the problem that can be evaluated using the cross-sectional dimensions and other data, are defined as:

Cross-sectional area:	$A = (ht_w + 2bt_f), \text{ m}^2$
Moment of inertia:	$I = \frac{1}{12}t_w h^3 + \frac{2}{3}bt_f^3 + \frac{1}{2}bt_f h(h + 2t_f), \text{ m}^4$
Uniform load for the girder:	$w = (19 + 77A), \text{ kN/m}$
Bending moment:	$M = \frac{L}{8}(2P_m + wL), \text{ kN-m}$
Bending stress:	$\sigma = \frac{M}{1000I}(0.5h + t_f), \text{ MPa}$
Flange buckling stress limit:	$\sigma_f = 72,845\left(\frac{t_f}{b}\right)^2, \text{ MPa}$
Web crippling stress limit:	$\sigma_w = 3,648,276\left(\frac{t_w}{h}\right)^2, \text{ MPa}$
Shear force:	$S = 0.5(P_s + wL), \text{ kN}$
Deflection:	$D = \frac{L^3}{384 \times 10^6 EI}(8P_m + 5wL), \text{ m}$
Average shear stress:	$\tau = \frac{S}{1000ht_w}, \text{ MPa}$

Identification/Definition of Design Variables Cross-sectional dimensions of the plate girder are treated as four design variables for the problem:

- h = web height, m
- b = flange width, m
- t_f = flange thickness, m
- t_w = web thickness, m

Identification of Criterion To Be Optimized The objective is to minimize the material volume of the girder:

$$\text{Vol} = AL = (ht_w + 2bt_f)L, \text{ m}^3 \quad (\text{a})$$

Identification of Constraints The following constraints for the plate girder are defined:

- Bending stress: $\sigma \leq \sigma_a$ (b)
- Flange buckling: $\sigma \leq \sigma_f$ (c)
- Web crippling: $\sigma \leq \sigma_w$ (d)
- Shear stress: $\tau \leq \tau_a$ (e)
- Deflection: $D \leq D_a$ (f)
- Fatigue stress: $\sigma \leq \frac{1}{2}\sigma_f$ (g)
- Size constraints: $0.30 \leq h \leq 2.5, 0.30 \leq b \leq 2.5$
 $0.01 \leq t_f \leq 0.10, 0.01 \leq t_w \leq 0.10$ (h)

Note that the lower and upper limits on the design variables have been specified arbitrarily in the present example. In practice, appropriate values for the given design problem will have to be specified based on the available plate sizes. It is important to note that constraints of Eqs. (b) to (g) can be written explicitly in terms of the design variables h , b , t_f , and t_w by substituting into them expressions for all the dependent variables. However, there are many applications where it is not possible or convenient to eliminate the dependent variables to obtain explicit expressions for all the functions of the optimization problem in terms of the design variables alone. In such cases, the dependent variables must be kept in the problem formulation and treated in the solution process. In addition, use of dependent variables makes it easier to read and debug the program that contains the problem formulation.

Spreadsheet Layout Layout of the spreadsheet for solution of KKT optimality conditions, linear programming problems, and unconstrained problems was explained earlier in Chapters 4, 6, and 8. As noted there, Solver is an “Add-in” to Microsoft Excel; if it does not appear under the “Tools” menu, it can be easily installed. Figure 10-15 shows the layout of the spreadsheet showing formulas for the plate girder design problem in various cells. The spreadsheet can be organized in any convenient way. The main requirements are that the cells containing objective and constraint functions and the design variables be clearly identified. For the present problem the spreadsheet is organized into five distinct blocks. The first block contains information about the design variables. Symbols for the variables and their upper and lower limits are defined. The cells containing the starting values for the variables are identified as D3 to D6. These are the cells that are updated during the solution process. Also

	A	B	C	D	E	F
1	Plate Girder Design					
2	1. Design variable name	Lower limit	Symbol	Value	Upper limit	Units
3	web height	0.3	h	1	2.5	m
4	flange width	0.3	b	1	2.5	m
5	flange thickness	0.01	tf	0.1	0.1	m
6	web thickness	0.01	tw	0.1	0.1	m
7						
8	2. Parameter name	Symbol	Value			Units
9	Span length	L	25			m
10	Modulus of elasticity	E	210			GPa
11	Yield stress	sigma_y	262			MPa
12	Allowable fatigue stress	sigma_t	255			MPa
13	Concentrated load for moment	Pm	104			kN
14	Concentrated load for shear	Ps	155			kN
15	Live load impact factor	LLIF	=1+50(L+125)			none
16						
17	3. Dependent Variable name	Symbol	Equation			Units
18	Cross sectional area	A	=h*tw+2*b*tf			m ²
19	Moment of inertia	I	=(1/12)*tw*h ³ +(2/3)*b*tf ³ +(1/2)*b*tf*h*(h+2*tf)			m ⁴
20	Uniform load	w	=19+77*A			kN/m
21	Bending moment	M	=L*(2*Pm+w*L)/8			kN-m
22	Bending stress	sigma	=M*(h/2+tf)/(1000*I)			MPa
23	Shear force	S	=(Ps+w*L)/2			kN
24	Deflection	D	=L ³ *(8*Pm+5*w*L)/(384*E*I*1000000)			m
25	Average shear stress	tau	=S/(1000*h*tw)			MPa
26						
27	4. Objective Function name	Symbol	Equation			Units
28	Volume of material	Vol	=A*L			m ³
29						
30	5. Constraints	Value/Eq.	</>=	Value/Eq.		Name
31	Bending stress	=sigma	<	=0.55*sigma_y		Allowable bending stress
32	Bending stress	=sigma	<	=72845*(tf/b)^2		Flange buckling limit
33	Bending stress	=sigma	<	=3648276*(tw/h)^2		Web crippling limit
34	Shear stress	=tau	<	=0.33*sigma_y		Allowable shear stress
35	Deflection	=D	<	=L/800		Allowable deflection
36	Bending stress	=sigma	<	=sigma_t/2		Allowable fatigue stress

FIGURE 10-15 Layout of the spreadsheet for plate girder design problem.

since these cells are used in all expressions, they are given real names, such as h, b, tf, tw. This is done by using the “Insert/Name” command. The second block defines various data and parameters for the problem. Material properties, loading data, and span length are defined. Equations for the dependent variables are entered in cells C18 to C25 in block 3. Although it is not necessary to include them (because they can be incorporated explicitly into the constraint and objective function formulas), it can be very useful. First they simplify the formulation of the constraint expressions, reducing algebraic manipulation errors. Second, they provide a check of these intermediate quantities for debugging purposes and for information feedback. Block 4 identifies the cell for the objective function, cell C28. Block 5 contains the information about the constraints. Cells B31 to B36 contain the left sides and cells D31 to D36 contain the right sides of the constraints. Constraints are implemented in Excel by relating two cells through an inequality (\leq or \geq) or an equality ($=$) relationship. This is defined in the Solver dialog box, which is described next. Although many of the quantities appearing in the constraint section also appear elsewhere in the spreadsheet, they are simply references to other cells in the variables section and the parameters section of the spreadsheet (see formulas in Fig. 10-15). This way, the only cells that need to be modified during a “what-if” analysis are those in the independent variable section or the parameters section. The constraints are automatically updated to reflect the changes.

Solver Dialog Box Once the spreadsheet has been created, the next step is to define the optimization problem for Solver. Figure 10-16 shows a screen shot of the Solver dialog box. The objective function cell is entered as the “Target Cell,” which is to be minimized. The independent design variables are identified next under the “By Changing Cells:” heading. A range of cells has been entered here, but individual cells, separated by commas could be entered instead. Finally the constraints are entered under the “Subject to the Constraints:”

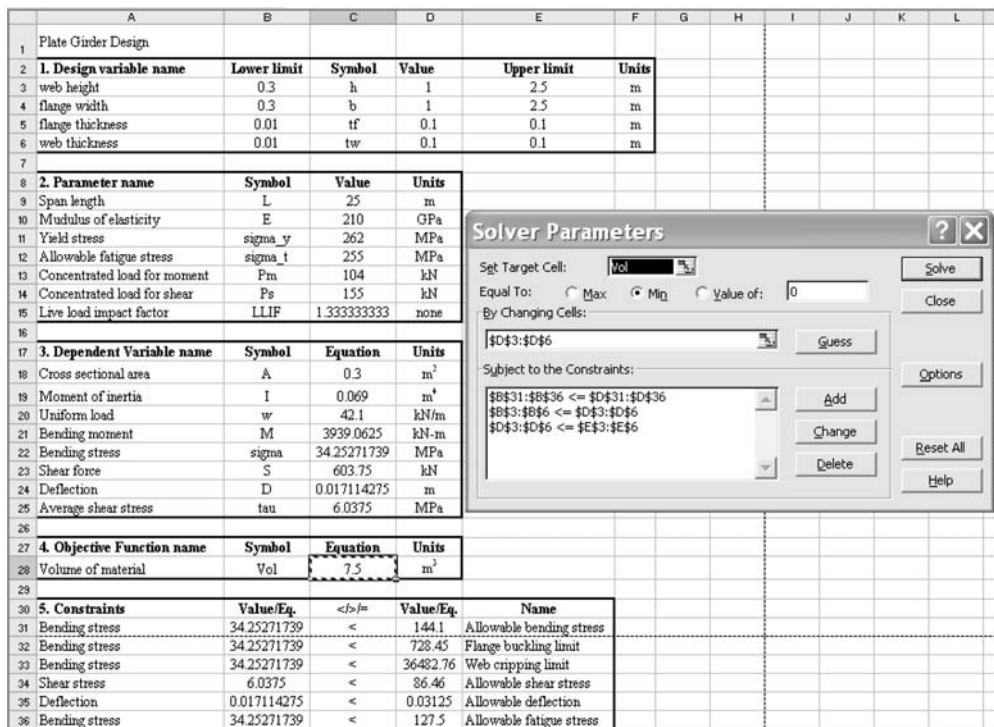


FIGURE 10-16 Solver dialog box and spreadsheet for plate girder design problem.

heading. The constraints include not only those identified in the constraints section of the spreadsheet but also the bounds on the design variables.

Solution Once the problem has been defined in the “Solver Dialog Box,” clicking Solve button initiates the optimization process. Once Solver has found the solution, the design variable cells, D3 to D6, dependent variable cells, C18 to C25, and the constraint function cells, B31 to B36 and D31 to D36 are updated using the optimum values of the design variables. Solver also generates three reports in separate worksheets, “Answer, Sensitivity, Limits” (as explained in Chapter 6). The Lagrange multipliers and constraint activity can be recovered from these reports. The solution is obtained as follows: $h = 2.0753$ m, $b = 0.3960$ m, $t_f = 0.0156$ m, $t_w = 0.0115$ m, Vol = 0.90563 m³. The flange buckling, web crippling, and deflection constraints are active at the optimum point.

It is important to note that once a design problem has been formulated and coupled to an optimization software, such as Excel, variations in the operating environment and other conditions for the problem can be investigated in a very short time. “What if” type questions can be investigated and *insights* about behavior of the system can be gained. For example the problem can be solved for the following conditions in a short time:

1. What happens if deflection or web crippling constraint is omitted from the formulation?
2. What if the span length is changed?
3. What if some material properties change?
4. What if a design variable is assigned a fixed value?
5. What if bounds on the variables are changed?

Exercises for Chapter 10

Section 10.1 Basic Concepts and Ideas

10.1 Answer True or False.

1. The basic numerical iterative philosophy for solving constrained and unconstrained problems is the same.
2. Step size determination is a one-dimensional problem for unconstrained problems.
3. Step size determination is a multidimensional problem for constrained problem.
4. An inequality constraint $g_i(\mathbf{x}) \leq 0$ is violated at $\mathbf{x}^{(k)}$ if $g_i(\mathbf{x}^{(k)}) > 0$.
5. An inequality constraint $g_i(\mathbf{x}) \leq 0$ is active at $\mathbf{x}^{(k)}$ if $g_i(\mathbf{x}^{(k)}) > 0$.
6. An equality constraint $h_i(\mathbf{x}) = 0$ is violated at $\mathbf{x}^{(k)}$ if $h_i(\mathbf{x}^{(k)}) > 0$.
7. An equality constraint is always active at the optimum.
8. In constrained optimization problems, search direction is found using the cost gradient only.
9. In constrained optimization problems, search direction is found using the constraint gradients only.
10. In constrained problems, the descent function is used to calculate the search direction.
11. In constrained problems, the descent function is used to calculate a feasible point.
12. Cost function can be used as a descent function in constrained problems.
13. One-dimensional search on a descent function is needed for convergence of algorithms.

14. A robust algorithm guarantees convergence.
15. A feasible set must be closed and bounded to guarantee convergence of algorithms.
16. A constraint $x_1 + x_2 \leq -2$ can be normalized as $(x_1 + x_2)/(-2) \leq 1.0$.
17. A constraint $x_1^2 + x_2^2 \leq 9$ is active at $x_1 = 3$ and $x_2 = 3$.

Section 10.2 Linearization of the Constrained Problem

10.2 *Answer True or False.*

1. Linearization of cost and constraint functions is a basic step for solving nonlinear optimization problems.
2. General constrained problems cannot be solved by solving a sequence of linear programming subproblems.
3. In general, the linearized subproblem without move limits may be unbounded.
4. The sequential linear programming method for general constrained problems is guaranteed to converge.
5. Move limits are essential in the sequential linear programming procedure.
6. Equality constraints can be treated in the sequential linear programming algorithm.

Formulate the following design problems, transcribe them into the standard form, create a linear approximation at the given point, and plot the linearized subproblem and the original problem on the same graph.

- 10.3 Beam design problem formulated in Section 3.8 at the point $(b, d) = (250, 300)$ mm.
- 10.4 Tubular column design problem formulated in Section 2.7 at the point $(R, t) = (12, 4)$ cm. Let $P = 50$ kN, $E = 210$ GPa, $l = 500$ cm, $\sigma_a = 250$ MPa, and $\rho = 7850$ kg/m³.
- 10.5 Wall bracket problem formulated in Section 4.7.1 at the point $(A_1, A_2) = (150, 150)$ cm².
- 10.6 Exercise 2.1 at the point $h = 12$ m, $A = 4000$ m².
- 10.7 Exercise 2.3 at the point $(R, H) = (6, 15)$ cm.
- 10.8 Exercise 2.4 at the point $R = 2$ cm, $N = 100$.
- 10.9 Exercise 2.5 at the point $(W, D) = (100, 100)$ m.
- 10.10 Exercise 2.9 at the point $(r, h) = (6, 16)$ cm.
- 10.11 Exercise 2.10 at the point $(b, h) = (5, 10)$ m.
- 10.12 Exercise 2.11 at the point, width = 5 m, depth = 5 m, and height = 5 m.
- 10.13 Exercise 2.12 at the point $D = 4$ m and $H = 8$ m.
- 10.14 Exercise 2.13 at the point $w = 10$ m, $d = 10$ m, $h = 4$ m.
- 10.15 Exercise 2.14 at the point $P_1 = 2$ and $P_2 = 1$.

Section 10.3 Sequential Linear Programming Algorithm

Complete one iteration of the sequential linear programming algorithm for the following problems (try 50 percent move limits and adjust them if necessary).

- 10.16 Beam design problem formulated in Section 3.8 at the point $(b, d) = (250, 300)$ mm.

- 10.17 Tubular column design problem formulated in Section 2.7 at the point $(R, t) = (12, 4)$ cm. Let $P = 50$ kN, $E = 210$ GPa, $l = 500$ cm, $\sigma_a = 250$ MPa, and $\sigma = 7850$ kg/m³.
- 10.18 Wall bracket problem formulated in Section 4.7.1 at the point $(A_1, A_2) = (150, 150)$ cm².
- 10.19 Exercise 2.1 at the point $h = 12$ m, $A = 4000$ m².
- 10.20 Exercise 2.3 at the point $(R, H) = (6, 15)$ cm.
- 10.21 Exercise 2.4 at the point $R = 2$ cm, $N = 100$.
- 10.22 Exercise 2.5 at the point $(W, D) = (100, 100)$ m.
- 10.23 Exercise 2.9 at the point $(r, h) = (6, 16)$ cm.
- 10.24 Exercise 2.10 at the point $(b, h) = (5, 10)$ m.
- 10.25 Exercise 2.11 at the point, width = 5 m, depth = 5 m, and height = 5 m.
- 10.26 Exercise 2.12 at the point $D = 4$ m and $H = 8$ m.
- 10.27 Exercise 2.13 at the point $w = 10$ m, $d = 10$ m, $h = 4$ m.
- 10.28 Exercise 2.14 at the point $P_1 = 2$ and $P_2 = 1$.

Section 10.4 Quadratic Programming Subproblem

Solve the following QP problems using KKT optimality conditions.

- 10.29 Minimize $f(\mathbf{x}) = (x_1 - 3)^2 + (x_2 - 3)^2$
 subject to $x_1 + x_2 \leq 5$
 $x_1, x_2 \geq 0$
- 10.30 Minimize $f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 - 1)^2$
 subject to $x_1 + 2x_2 \leq 6$
 $x_1, x_2 \geq 0$
- 10.31 Minimize $f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 - 1)^2$
 subject to $x_1 + 2x_2 \leq 2$
 $x_1, x_2 \geq 0$
- 10.32 Minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - x_1x_2 - 3x_1$
 subject to $x_1 + x_2 \leq 3$
 $x_1, x_2 \geq 0$
- 10.33 Minimize $f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 - 1)^2 - 2x_2 + 2$
 subject to $x_1 + x_2 \leq 4$
 $x_1, x_2 \geq 0$
- 10.34 Minimize $f(\mathbf{x}) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8x_1$
 subject to $x_1 + x_2 = 4$
 $x_1, x_2 \geq 0$
- 10.35 Minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - 2x_1 - 2x_2$
 subject to $x_1 + x_2 - 4 = 0$
 $x_1 - x_2 - 2 = 0$
 $x_1, x_2 \geq 0$

- 10.36 Minimize $f(\mathbf{x}) = 4x_1^2 + 3x_2^2 - 5x_1x_2 - 8x_1$
subject to $x_1 + x_2 \leq 4$
 $x_1, x_2 \geq 0$
- 10.37 Minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - 4x_1 - 2x_2$
subject to $x_1 + x_2 \geq 4$
 $x_1, x_2 \geq 0$
- 10.38 Minimize $f(\mathbf{x}) = 2x_1^2 + 6x_1x_2 + 9x_2^2 - 18x_1 + 9x_2$
subject to $x_1 - 2x_2 \leq 10$
 $4x_1 - 3x_2 \leq 20$
 $x_1, x_2 \geq 0$
- 10.39 Minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - 2x_1 - 2x_2$
subject to $x_1 + x_2 - 4 \leq 0$
 $2 - x_1 \leq 0$
 $x_1, x_2 \geq 0$
- 10.40 Minimize $f(\mathbf{x}) = 2x_1^2 + 2x_2^2 + x_3^2 + 2x_1x_2 - x_1x_3 - 0.8x_2x_3$
subject to $1.3x_1 + 1.2x_2 + 1.1x_3 \geq 1.15$
 $x_1 + x_2 + x_3 = 1$
 $x_1 \leq 0.7$
 $x_2 \leq 0.7$
 $x_3 \leq 0.7$
 $x_1, x_2, x_3 \geq 0$

For the following problems, obtain the quadratic programming subproblem, plot it on a graph, obtain the search direction for the subproblem, and show the search direction on the graphical representation of the original problem.

- 10.41 Beam design problem formulated in Section 3.8 at the point $(b, d) = (250, 300)$ mm.
- 10.42 Tubular column design problem formulated in Section 2.7 at the point $(R, t) = (12, 4)$ cm. Let $P = 50$ kN, $E = 210$ GPa, $l = 500$ cm, $\sigma_a = 250$ MPa, and $\rho = 7850$ kg/m³.
- 10.43 Wall bracket problem formulated in Section 4.7.1 at the point $(A_1, A_2) = (150, 150)$ cm².
- 10.44. Exercise 2.1 at the point $h = 12$ m, $A = 4000$ m².
- 10.45 Exercise 2.3 at the point $(R, H) = (6, 15)$ cm.
- 10.46 Exercise 2.4 at the point $R = 2$ cm, $N = 100$.
- 10.47 Exercise 2.5 at the point $(W, D) = (100, 100)$ m.
- 10.48 Exercise 2.9 at the point $(r, h) = (6, 16)$ cm.
- 10.49 Exercise 2.10 at the point $(b, h) = (5, 10)$ m.
- 10.50 Exercise 2.11 at the point, width = 5 m, depth = 5 m, and height = 5 m.
- 10.51 Exercise 2.12 at the point $D = 4$ m and $H = 8$ m.
- 10.52 Exercise 2.13 at the point $w = 10$ m, $d = 10$ m, $h = 4$ m.
- 10.53 Exercise 2.14 at the point $P_1 = 2$ and $P_2 = 1$.

Section 10.5 Constrained Steepest Descent Method

10.54 *Answer True or False.*

1. The constrained steepest descent (CSD) method, when there are active constraints, is based on using the cost function gradient as the search direction.
2. The constrained steepest descent method solves two subproblems: the search direction and step size determination.
3. The cost function is used as the descent function in the CSD method.
4. The QP subproblem in the CSD method is strictly convex.
5. The search direction, if one exists, is unique for the QP subproblem in the CSD method.
6. Constraint violations play no role in step size determination in the CSD method.
7. Lagrange multipliers of the subproblem play a role in step size determination in the CSD method.
8. Constraints must be evaluated during line search in the CSD method.

For the following problems, complete one iteration of the constrained steepest descent method for the given starting point (let $R_0 = 1$, and determine an approximate step size using the golden section method).

- 10.55 Beam design problem formulated in Section 3.8 at the point $(b, d) = (250, 300)$ mm.
- 10.56 Tubular column design problem formulated in Section 2.7 at the point $(R, t) = (12, 4)$ cm. Let $P = 50$ kN, $E = 210$ GPa, $l = 500$ cm, $\sigma_a = 250$ MPa, and $\rho = 7850$ kg/m³.
- 10.57 Wall bracket problem formulated in Section 4.7.1 at the point $(A_1, A_2) = (150, 150)$ cm².
- 10.58 Exercise 2.1 at the point $h = 12$ m, $A = 4000$ m².
- 10.59 Exercise 2.3 at the point $(R, H) = (6, 15)$ cm.
- 10.60 Exercise 2.4 at the point $R = 2$ cm, $N = 100$.
- 10.61 Exercise 2.5 at the point $(W, D) = (100, 100)$ m.
- 10.62 Exercise 2.9 at the point $(r, h) = (6, 16)$ cm.
- 10.63 Exercise 2.10 at the point $(b, h) = (5, 10)$ m.
- 10.64 Exercise 2.11 at the point, width = 5 m, depth = 5 m, and height = 5 m.
- 10.65 Exercise 2.12 at the point $D = 4$ m and $H = 8$ m.
- 10.66 Exercise 2.13 at the point $w = 10$ m, $d = 10$ m, $h = 4$ m.
- 10.67 Exercise 2.14 at the point $P_1 = 2$ and $P_2 = 1$.

Section 10.6 Engineering Design Optimization Using Excel Solver

Formulate and solve the following problems using Excel Solver or another software.

- | | | |
|----------------------|----------------------|----------------------|
| 10.68* Exercise 3.34 | 10.69* Exercise 3.35 | 10.70* Exercise 3.36 |
| 10.71* Exercise 3.50 | 10.72* Exercise 3.51 | 10.73* Exercise 3.52 |
| 10.74* Exercise 3.53 | 10.75* Exercise 3.54 | |

11 More on Numerical Methods for Constrained Optimum Design

Upon completion of this chapter, you will be able to:

- Use potential constraint strategy in numerical optimization algorithms for constrained problems
- Determine whether a software for nonlinear constrained optimization problems uses potential constraint strategy that is appropriate for most engineering applications
- Use approximate step size that is more efficient for constrained optimization methods
- Use quasi-Newton methods to solve constrained nonlinear optimization problems
- Explain basic ideas behind methods of feasible direction, gradient projection, and generalized reduced gradient

In Chapter 10, basic concepts and steps related to constrained optimization methods were presented and illustrated. In this chapter, we build upon those basic ideas and describe some concepts and methods that are more appropriate for practical applications. Topics such as inaccurate line search, constrained quasi-Newton methods, and potential constraint strategy to define the quadratic programming subproblem are discussed and illustrated. These topics usually should not be covered in an undergraduate course on optimum design or on first independent reading of the text.

For convenience of reference, the general constrained optimization problem treated in the previous chapter is restated as: find $\mathbf{x} = (x_1, \dots, x_n)$, a design variable vector of dimension n , to minimize a cost function $f = f(\mathbf{x})$ subject to equality constraints $h_i(\mathbf{x}) = 0$, $i = 1$ to p and inequality constraints $g_i(\mathbf{x}) \leq 0$, $i = 1$ to m .

11.1 Potential Constraint Strategy

To evaluate the search direction in numerical methods for constrained optimization, one needs to know the cost and constraint functions and their gradients. The numerical algorithms for constrained optimization can be classified based on whether gradients of all the constraints

or only a subset of them are required to define the search direction determination subproblem. The numerical algorithms that use gradients of only a subset of the constraints in the definition of this subproblem are said to use *potential constraint strategy*. To implement this strategy, a potential constraint index set needs to be defined, which is composed of active, ϵ -active, and violated constraints at the current iteration. At the k th iteration, we define a potential constraint index set I_k as follows:

$$I_k = [\{j \mid j = 1 \text{ to } p \text{ for equalities}\} \text{ and } \{i \mid g_i(\mathbf{x}^{(k)}) + \epsilon \geq 0, i = 1 \text{ to } m\}] \quad (11.1)$$

Note that the set I_k contains a list of constraints that satisfy the criteria given in Eq. (11.1); all the *equality constraints are always included in I_k by definition*. The main effect of using this strategy in an algorithm is on the efficiency of the entire iterative process. This is particularly true for large and complex applications where the evaluation of gradients of constraints is expensive. With the potential set strategy, gradients of only the constraints in the set I_k are calculated and used in defining the search direction determination subproblem. The original problem may have hundreds of constraints, but only a few may be in the potential set. Thus with this strategy, not only the number of gradient evaluations is reduced but also the dimension of the subproblem for the search direction is substantially reduced. This can result in additional saving in the computational effort. Therefore, *the potential set strategy is beneficial and should be used in practical applications of optimization*. Before using software to solve a problem, the designer should inquire whether the program uses the potential constraint strategy. Example 11.1 illustrates determination of a potential constraint set for an optimization problem.

EXAMPLE 11.1 Determination of Potential Constraint Set

Consider the following six constraints:

$$2x_1^2 + x_2 \leq 36; \quad x_1 \geq 60x_2; \quad x_2 \leq 10; \quad x_2 + 2 \geq 0; \quad x_1 \leq 10; \quad x_1 \geq 0$$

Let $\mathbf{x}^{(k)} = (-4.5, -4.5)$ and $\epsilon = 0.1$. Form the potential constraint index set I_k of Eq. (11.1).

Solution. After normalization and conversion to the standard form, the constraints are given as

$$g_1 = \frac{1}{18}x_1^2 + \frac{1}{36}x_2 - 1 \leq 0, \quad g_2 = \frac{1}{100}(-x_1 + 60x_2) \leq 0 \quad (a)$$

$$g_3 = \frac{1}{10}x_2 - 1 \leq 0, \quad g_4 = -\frac{1}{2}x_2 - 1 \leq 0 \quad (b)$$

$$g_5 = \frac{1}{10}x_1 - 1 \leq 0, \quad g_6 = -x_1 \leq 0 \quad (c)$$

Since the second constraint does not have a constant in its expression, the constraint is divided by 100 to get a percent value of the constraint. Evaluating the constraints at the given point $(-4.5, -4.5)$, we obtain

$$g_1 = \frac{1}{18}(-4.5)^2 + \frac{1}{36}(-4.5) - 1.0 = 0 \text{ (active)} \quad (d)$$

$$g_2 = \frac{1}{100}[-(-4.5) + 60(-4.5)] = -2.655 < 0 \text{ (inactive)} \quad (e)$$

$$g_3 = \frac{-4.5}{10} - 1.0 = -1.45 < 0 \text{ (inactive)} \quad (f)$$

$$g_4 = -\frac{1}{2}(-4.5) - 1.0 = 1.25 > 0 \text{ (violated)} \quad (g)$$

$$g_5 = \frac{1}{10}(-4.5) - 1.0 = -1.45 < 0 \text{ (inactive)} \quad (h)$$

$$g_6 = -(-4.5) = 4.5 > 0 \text{ (violated)} \quad (i)$$

Therefore, we see that g_1 is active (also ε - active); g_4 and g_6 are violated; and g_2 , g_3 , and g_5 are inactive. Thus, $I_k = \{1, 4, 6\}$.

It is important to note that a numerical algorithm using the potential constraint strategy must be proved to be convergent. The potential set strategy has been incorporated into the CSD algorithm of Chapter 10; that algorithm has been proved to be convergent to a local minimum point starting from any point. Note that the elements of the index set depend on the value of ε used in Eq. (11.1). Also the search direction with different index sets can be different, giving a different path to the optimum point. Example 11.2 calculates the search directions with and without the potential set strategy and shows that they are different.

EXAMPLE 11.2 Search Direction with and without Potential Constraint Strategy

Consider the design optimization problem:

$$\text{minimize } f(\mathbf{x}) = x_1^2 - 3x_1x_2 + 4.5x_2^2 - 10x_1 - 6x_2 \quad (a)$$

subject to the constraints

$$x_1 - x_2 \leq 3, \quad x_1 + 2x_2 \leq 12, \quad \text{and} \quad x_1, x_2 \geq 0 \quad (b)$$

At the point (4, 4) calculate the search directions with and without the potential set strategy. Use $\varepsilon = 0.1$.

Solution. Writing constraints in the standard normalized form, we get

$$g_1 = \frac{1}{3}(x_1 - x_2) - 1 \leq 0, \quad g_2 = \frac{1}{12}(x_1 + 2x_2) - 1 \leq 0 \quad g_3 = -x_1 \leq 0 \quad g_4 = -x_2 \leq 0 \quad (c)$$

At the point (4, 4), functions and their gradients are calculated as

$$f(4,4) = -24, \quad \mathbf{c} = \nabla f = (2x_1 - 3x_2 - 10, -3x_1 + 9x_2 - 6) = (-14, 18) \quad (\text{d})$$

$$g_1(4,4) = -1 < 0 \text{ (inactive)}, \quad \mathbf{a}^{(1)} = \nabla g_1 = \left(\frac{1}{3}, -\frac{1}{3}\right) \quad (\text{e})$$

$$g_2(4,4) = 0 \text{ (active)}, \quad \mathbf{a}^{(2)} = \nabla g_2 = \left(\frac{1}{12}, \frac{1}{6}\right) \quad (\text{f})$$

$$g_3(4,4) = -4 < 0 \text{ (inactive)}, \quad \mathbf{a}^{(3)} = \nabla g_3 = (-1, 0) \quad (\text{g})$$

$$g_4(4,4) = -4 < 0 \text{ (inactive)}, \quad \mathbf{a}^{(4)} = \nabla g_4 = (0, -1) \quad (\text{h})$$

When the potential constraint strategy is not used, the QP subproblem of Eqs. (10.25) and (10.26) is defined as

minimize

$$\tilde{f} = -14d_1 + 18d_2 + \frac{1}{2}(d_1^2 + d_2^2) \quad (\text{i})$$

subject to

$$\begin{bmatrix} \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{12} & \frac{1}{6} \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 4 \\ 4 \end{bmatrix} \quad (\text{j})$$

Solution of the problem using the KKT necessary conditions of Theorem 4.6 is given as $\mathbf{d} = (-0.5, -3.5)$, $\mathbf{u} = (43.5, 0, 0, 0)$. If we use the potential constraint strategy, the index set I_k is defined as $I_k = \{2\}$, that is only the second constraint needs to be considered in defining the QP subproblem. With this strategy, the QP subproblem is defined as

minimize

$$\tilde{f} = -14d_1 + 18d_2 + \frac{1}{2}(d_1^2 + d_2^2) \quad (\text{k})$$

subject to

$$\frac{1}{12}d_1 + \frac{1}{6}d_2 \leq 0 \quad (\text{l})$$

Solution of this problem using the KKT necessary conditions is given as $\mathbf{d} = (14, -18)$, $u = 0$. Thus it is seen that the search directions determined by the two subproblems are quite different. The path to the optimum solution and the computational effort will also be quite different.

11.2 Quadratic Programming Problem

A quadratic programming (QP) problem has a quadratic cost function and linear constraints. Such problems are encountered in many real-world applications. In addition, many general nonlinear programming algorithms require solution of a quadratic programming subproblem at each iteration. As seen in Eqs. (10.25) and (10.26), the QP subproblem is obtained when a nonlinear problem is linearized and a quadratic step size constraint is imposed. It is important to solve the QP subproblem efficiently so that large-scale problems can be treated. Thus, it is not surprising that substantial research effort has been expended in developing and evaluating many algorithms for solving QP problems (Gill *et al.*, 1981; Luenberger, 1984). Also, several commercially available software packages are available for solving QP problems, e.g., MATLAB, QPSOL (Gill *et al.*, 1984), VE06A (Hopper, 1981), and E04NAF (NAG, 1984). Some of the available LP codes also have an option to solve QP problems (Schrage, 1981).

To give a flavor of the calculations needed to solve QP problems, we shall describe a method that is a simple extension of the *Simplex method*. Many other methods are available that can be considered as variations of that procedure in numerical implementation details.

11.2.1 Definition of QP Problem

Let us define a general QP problem as follows:

$$\text{minimize } q(\mathbf{x}) = \mathbf{c}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x} \quad (11.2)$$

subject to linear equality and inequality constraints

$$\mathbf{N}^T \mathbf{x} = \mathbf{e} \quad (11.3)$$

$$\mathbf{A}^T \mathbf{x} \leq \mathbf{b} \quad (11.4)$$

and nonnegativity of the variables

$$\mathbf{x} \geq \mathbf{0} \quad (11.5)$$

where $\mathbf{c} = n$ dimensional constant vector

$\mathbf{x} = n$ dimensional vector of unknowns

$\mathbf{b} = m$ dimensional constant vector

$\mathbf{e} = p$ dimensional constant vector

$\mathbf{H} = n \times n$ constant Hessian matrix

$\mathbf{A} = n \times m$ constant matrix

$\mathbf{N} = n \times p$ constant matrix

Note that all the linear inequality constraints are expressed in the “ \leq form.” This is needed because we shall use KKT necessary conditions of Section 4.4, which require this form. Note also that if the matrix \mathbf{H} is positive semidefinite, the QP problem is convex, so any solution (if one exists) represents a global minimum point (which need not be unique). Further, if the matrix \mathbf{H} is positive definite, the problem is strictly convex. Therefore, the problem has a unique global solution (if one exists). We shall assume that the matrix \mathbf{H} is at least positive semidefinite. This is not an unreasonable assumption in practice as many applications satisfy it. For example, in the QP subproblem of Eqs. (10.25) and (10.26), $\mathbf{H} = \mathbf{I}$ (an identity matrix),

so the Hessian is actually positive definite. Note also that the variables \mathbf{x} are required to be nonnegative in Eq. (11.5). Variables that are free in sign can be easily treated by the method described in Section 6.1.

11.2.2 KKT Necessary Conditions for the QP Problem

A procedure for solving the QP problem of Eqs. (11.2) to (11.5) is to first write the KKT necessary conditions of Section 4.4, and then to transform them into a form that can be treated by Phase I of the Simplex method of Section 6.4. To write the necessary conditions, we introduce slack variables \mathbf{s} for Inequalities (11.4) and transform them to equalities as

$$\mathbf{A}^T \mathbf{x} + \mathbf{s} = \mathbf{b}; \quad \text{with} \quad \mathbf{s} \geq \mathbf{0} \quad (11.6)$$

Or, the slack variable for the j th inequality in Eq. (11.4) can be expressed using Eq. (11.6) as

$$s_j = b_j - \sum_{i=1}^n a_{ij} x_i \quad (\mathbf{s} = \mathbf{b} - \mathbf{A}^T \mathbf{x}) \quad (11.7)$$

Note the nonnegativity constraints of Eq. (11.5) (when expressed in the standard form $-\mathbf{x} \leq \mathbf{0}$) do not need slack variables because $x_i \geq 0$ itself is a slack variable. Let us now define the Lagrange function of Eq. (4.46a) for the QP problem as

$$L = \mathbf{c}^T \mathbf{x} + 0.5 \mathbf{x}^T \mathbf{H} \mathbf{x} + \mathbf{u}^T (\mathbf{A}^T \mathbf{x} + \mathbf{s} - \mathbf{b}) - \xi^T \mathbf{x} + \mathbf{v}^T (\mathbf{N}^T \mathbf{x} - \mathbf{e})$$

where \mathbf{u} , \mathbf{v} , and ξ are the Lagrange multiplier vectors for the inequality constraints of Eq. (11.4) or (11.7), equality constraints of Eq. (11.3), and the nonnegativity constraints ($-\mathbf{x} \leq \mathbf{0}$), respectively. The KKT necessary conditions give

$$\frac{\partial L}{\partial \mathbf{x}} = \mathbf{c} + \mathbf{H} \mathbf{x} + \mathbf{A} \mathbf{u} - \xi + \mathbf{N} \mathbf{v} = \mathbf{0} \quad (11.8)$$

$$\mathbf{A}^T \mathbf{x} + \mathbf{s} - \mathbf{b} = \mathbf{0} \quad (11.9)$$

$$\mathbf{N}^T \mathbf{x} - \mathbf{e} = \mathbf{0} \quad (11.10)$$

$$u_i s_i = 0; \quad i = 1 \text{ to } m \quad (11.11)$$

$$\xi_i x_i = 0; \quad i = 1 \text{ to } n \quad (11.12)$$

$$s_i, u_i \geq 0 \text{ for } i = 0 \text{ to } m; \quad \xi_i \geq 0 \text{ for } i = 1 \text{ to } n \quad (11.13)$$

These conditions need to be solved for \mathbf{x} , \mathbf{u} , \mathbf{v} , \mathbf{s} , and ξ .

11.2.3 Transformation of KKT Conditions

Before discussing the method for solution of KKT conditions, we shall transform them into a more compact form in this subsection. Since the Lagrange multipliers \mathbf{v} for the equality constraints are free in sign, we may decompose them as

$$\mathbf{v} = \mathbf{y} - \mathbf{z} \text{ with } \mathbf{y}, \mathbf{z} \geq \mathbf{0} \quad (11.14)$$

Now, writing Eqs. (11.8), (11.9), and (11.10) into a matrix form, we get

$$\begin{bmatrix} \mathbf{H} & \mathbf{A} & -\mathbf{I}_{(n)} & \mathbf{0}_{(n \times m)} & \mathbf{N} & -\mathbf{N} \\ \mathbf{A}^T & \mathbf{0}_{(m \times m)} & \mathbf{0}_{(m \times n)} & \mathbf{I}_{(m)} & \mathbf{0}_{(m \times p)} & \mathbf{0}_{(m \times p)} \\ \mathbf{N}^T & \mathbf{0}_{(p \times m)} & \mathbf{0}_{(p \times n)} & \mathbf{0}_{(p \times m)} & \mathbf{0}_{(p \times p)} & \mathbf{0}_{(p \times p)} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ \xi \\ \mathbf{s} \\ \mathbf{y} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} -\mathbf{c} \\ \mathbf{b} \\ \mathbf{e} \end{bmatrix} \quad (11.15)$$

where $\mathbf{I}_{(n)}$ and $\mathbf{I}_{(m)}$ are $n \times n$ and $m \times m$ identity matrices respectively, and $\mathbf{0}$ are zero matrices of the indicated order. In a compact matrix notation, Eq. (11.15) becomes

$$\mathbf{B}\mathbf{X} = \mathbf{D} \quad (11.16)$$

where matrix \mathbf{B} and vectors \mathbf{X} and \mathbf{D} are identified from Eq. (11.15) as

$$\mathbf{B} = \begin{bmatrix} \mathbf{H} & \mathbf{A} & -\mathbf{I}_{(n)} & \mathbf{0}_{(n \times m)} & \mathbf{N} & -\mathbf{N} \\ \mathbf{A}^T & \mathbf{0}_{(m \times m)} & \mathbf{0}_{(m \times n)} & \mathbf{I}_{(m)} & \mathbf{0}_{(m \times p)} & \mathbf{0}_{(m \times p)} \\ \mathbf{N}^T & \mathbf{0}_{(p \times m)} & \mathbf{0}_{(p \times n)} & \mathbf{0}_{(p \times m)} & \mathbf{0}_{(p \times p)} & \mathbf{0}_{(p \times p)} \end{bmatrix}_{[(n+m+p) \times (2n+2m+2p)]} \quad (11.17)$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ \xi \\ \mathbf{s} \\ \mathbf{y} \\ \mathbf{z} \end{bmatrix}_{(2n+2m+2p)} \quad \mathbf{D} = \begin{bmatrix} -\mathbf{c} \\ \mathbf{b} \\ \mathbf{e} \end{bmatrix}_{(n+m+p)} \quad (11.18)$$

The KKT conditions are now reduced to finding \mathbf{X} as a solution of the linear system in Eq. (11.16) subject to the constraints of Eqs. (11.11) to (11.13). In the new variables X_i the complementary slackness conditions of Eqs. (11.11) and (11.12), reduce to

$$X_i X_{n+m+i} = 0; \quad i = 1 \text{ to } (n+m) \quad (11.19)$$

and the nonnegativity conditions of Eq. (11.13) reduce to

$$X_i \geq 0; \quad i = 1 \text{ to } (2n+2m+2p) \quad (11.20)$$

11.2.4 Simplex Method for Solving QP Problem

A solution of the linear system in Eq. (11.16) that satisfies the complementary slackness of Eq. (11.19) and nonnegativity condition of Eq. (11.20) is a solution of the original QP problem. Note that the complementary slackness condition of Eq. (11.19) is nonlinear in the variables X_i . Therefore, it may appear that the Simplex method for LP cannot be used to solve Eq. (11.16). However, a procedure developed by Wolfe (1959) and refined by Hadley (1964) can be used to solve the problem. The procedure converges to a solution in the finite number of steps provided the matrix \mathbf{H} in Eq. (11.2) is positive definite. It can be further shown (Kunzi and Krelle, 1966, p. 123) that the method converges even when \mathbf{H} is positive semi-definite provided the vector \mathbf{c} in Eq. (11.2) is zero.

The method is based on Phase I of the Simplex procedure of Chapter 6 where we introduced an artificial variable for each equality constraint, defined an artificial cost function, and used it to determine an initial basic feasible solution. Following that procedure we introduce an artificial variable Y_i for each of the Eq. (11.16) as

$$\mathbf{B}\mathbf{X} + \mathbf{Y} = \mathbf{D} \quad (11.21)$$

where \mathbf{Y} is an $(n + m + p)$ dimensional vector. This way, we initially choose all X_i as non-basic and Y_j as basic variables. Note that all elements in \mathbf{D} must be nonnegative for the initial basic solution to be feasible. If any of the elements in \mathbf{D} are negative, the corresponding equation in Eq. (11.16) must be multiplied by -1 to have a nonnegative element on the right side. The artificial cost function for the problem is defined as

$$w = \sum_{i=1}^{n+m+p} Y_i \quad (11.22)$$

To use the Simplex procedure, we need to express the artificial cost function w in terms of nonbasic variables only. We eliminate basic variables Y_i from Eq. (11.22) by substituting Eq. (11.21) into it as

$$w = \sum_{i=1}^{n+m+p} D_i - \sum_{j=1}^{2(n+m+p)} \sum_{i=1}^{n+m+p} B_{ij} X_j = w_0 + \sum_{j=1}^{2(n+m+p)} C_j X_j \quad (11.23)$$

$$C_j = - \sum_{i=1}^{n+m+p} B_{ij} \quad \text{and} \quad w_0 = \sum_{i=1}^{n+m+p} D_i \quad (11.24)$$

Thus w_0 is the initial value of the artificial cost function and C_j is the initial relative cost coefficient obtained by adding the elements of the j th column of the matrix \mathbf{B} and changing its sign. Before we can use Phase I of the Simplex method, we need to develop a procedure to impose the complementary slackness condition of Eq. (11.19). The condition is satisfied if both X_i and X_{n+m+i} are not simultaneously basic. Or, if they are, then one of them has zero value (degenerate basic feasible solution). These conditions can easily be checked while determining the pivot element in the Simplex method.

It is useful to note here that a slightly different procedure to solve the KKT necessary conditions for the QP problem has been developed by Lemke (1965). It is known as the *complementary pivot method*. Numerical experiments have shown that method to be computationally more attractive than many other methods for solving QP problems when matrix \mathbf{H} is positive semidefinite (Ravindran and Lee, 1981). Example 11.3 illustrates use of the Simplex method to solve a QP problem.

EXAMPLE 11.3 Solution of QP Problem

$$\text{Minimize } f(\mathbf{x}) = (x_1 - 3)^2 + (x_2 - 3)^2 \quad (\text{a})$$

subject to

$$x_1 + x_2 \leq 4, \quad x_1 - 3x_2 = 1, \quad x_1, x_2 \geq 0 \quad (\text{b})$$

Solution. The cost function for the problem can be expanded as $f(\mathbf{x}) = x_1^2 - 6x_1 + x_2^2 - 6x_2 + 18$. We shall ignore the constant 18 in the cost function and minimize the following quadratic function expressed in the form of Eq. (11.2):

$$q(\mathbf{x}) = \begin{bmatrix} -6 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 0.5 \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (c)$$

From the foregoing equations, the following quantities can be identified:

$$\mathbf{c} = \begin{bmatrix} -6 \\ -6 \end{bmatrix}; \quad \mathbf{H} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}; \quad \mathbf{A} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \quad \mathbf{b} = [4]; \quad \mathbf{N} = \begin{bmatrix} 1 \\ -3 \end{bmatrix}; \quad \mathbf{e} = [1] \quad (d)$$

Using these quantities, matrix \mathbf{B} and vectors \mathbf{D} and \mathbf{X} of Eqs. (11.17) and (11.18) are identified as

$$\mathbf{B} = \left[\begin{array}{cc|cc|cc|cc|c} 2 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & \\ 0 & 2 & 1 & 0 & -1 & 0 & -3 & 3 & \\ \hline 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & \\ \hline 1 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & \end{array} \right] \quad \mathbf{D} = [6 \quad 6 \mid 4 \mid 1]^T \quad (e)$$

$$\mathbf{X} = [x_1 \quad x_2 \mid u_1 \mid \xi_1 \quad \xi_2 \mid s_1 \mid y_1 \quad z_1]^T$$

Table 11-1 shows the initial Simplex tableau as well as the four iterations to reach the optimum solution. Note that the relative cost coefficient C_j in the initial tableau is obtained by adding all the elements in the j th column and changing the sign of the sum. Also, the complementary slackness condition of Eq. (11.19) requires $X_1X_4 = 0$, $X_2X_5 = 0$, $X_3X_6 = 0$ implying that X_1 and X_4 , X_2 and X_5 , and X_3 and X_6 cannot be basic variables simultaneously. We impose these conditions while determining the pivots in Phase I of the Simplex procedure. After four iterations of the Simplex method, all the artificial variables are nonbasic and the artificial cost function is zero. Therefore, the optimum solution is given as

$$X_1 = \frac{13}{4}, \quad X_2 = \frac{3}{4}, \quad X_3 = \frac{3}{4}, \quad X_8 = \frac{3}{4} \quad (e)$$

$$X_4 = 0, \quad X_5 = 0, \quad X_6 = 0, \quad X_7 = 0$$

Using these values, the optimum solution for the original QP problem is recovered as

$$x_1 = \frac{13}{4}, \quad x_2 = \frac{3}{4}, \quad u_1 = \frac{3}{4}, \quad \xi_1 = 0, \quad \xi_2 = 0$$

$$s_1 = 0, \quad y_1 = 0, \quad z_1 = \frac{5}{4}, \quad v_1 = y_1 - z_1 = -\frac{5}{4} \quad (f)$$

$$f\left(\frac{13}{4}, \frac{3}{4}\right) = \frac{41}{8}$$

It can be verified that the solution satisfies all the KKT conditions for the problem.

Table 11-1 Simplex Solution Procedure for QP Problem of Example 11.2

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	Y_1	Y_2	Y_3	Y_4	D
Initial													
Y_1	2	0	1	-1	0	0	1	-1	1	0	0	0	6
Y_2	0	2	1	0	-1	0	-3	3	0	1	0	0	6
Y_3	1	1	0	0	0	1	0	0	0	0	1	0	4
Y_4	1	-3	0	0	0	0	0	0	0	0	0	1	1
	-4	0	-2	1	1	-1	2	-2	0	0	0	0	$w - 17$
First iteration													
Y_1	0	6	1	-1	0	0	1	-1	1	0	0	-2	4
Y_2	0	2	1	0	-1	0	-3	3	0	1	0	0	6
Y_3	0	4	0	0	0	1	0	0	0	0	1	-1	3
X_1	1	-3	0	0	0	0	0	0	0	0	0	1	1
	0	-12	-2	1	1	-1	2	-2	0	0	0	4	$w - 13$
Second iteration													
X_2	0	1	$\frac{1}{6}$	$-\frac{1}{6}$	0	0	$\frac{1}{6}$	$-\frac{1}{6}$	$\frac{1}{6}$	0	0	$-\frac{1}{3}$	$\frac{2}{3}$
Y_2	0	0	$\frac{2}{3}$	$-\frac{1}{3}$	-1	0	$-\frac{10}{3}$	$\frac{10}{3}$	$-\frac{1}{3}$	1	0	$-\frac{2}{3}$	$\frac{14}{3}$
Y_3	0	0	$-\frac{2}{3}$	$\frac{2}{3}$	0	1	$-\frac{2}{3}$	$\frac{2}{3}$	$-\frac{2}{3}$	0	1	$\frac{1}{3}$	$\frac{1}{3}$
X_1	1	0	$\frac{1}{2}$	$-\frac{1}{2}$	0	0	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	0	0	0	3
	0	0	0	-1	1	-1	4	-4	2	0	0	0	$w - 5$
Third iteration													
X_2	0	1	0	0	0	$\frac{1}{4}$	0	0	0	0	$\frac{1}{4}$	$-\frac{1}{4}$	$\frac{3}{4}$
Y_2	0	0	4	-3	-1	-5	0	0	3	1	-5	-1	3
X_8	0	0	-1	1	0	$\frac{3}{2}$	-1	1	-1	0	$\frac{3}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
X_1	1	0	0	0	0	$\frac{3}{4}$	0	0	0	0	$\frac{3}{4}$	$\frac{1}{4}$	$\frac{7}{4}$
	0	0	-4	3	1	5	0	0	-2	0	6	2	$w - 3$
Fourth iteration													
X_2	0	1	0	0	0	$\frac{1}{4}$	0	0	0	0	$\frac{1}{4}$	$-\frac{1}{4}$	$\frac{3}{4}$
X_3	0	0	1	$-\frac{3}{4}$	$-\frac{1}{4}$	$-\frac{5}{4}$	0	0	$\frac{3}{4}$	$\frac{1}{4}$	$-\frac{5}{4}$	$-\frac{1}{4}$	$\frac{3}{4}$
X_8	0	0	0	$\frac{1}{4}$	$-\frac{1}{4}$	$\frac{1}{4}$	-1	1	$-\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{5}{4}$
X_1	1	0	0	0	0	$\frac{3}{4}$	0	0	0	0	$\frac{3}{4}$	$\frac{1}{4}$	$\frac{13}{4}$
	0	0	0	0	0	0	0	1	1	1	1		$w - 0$

11.3 Approximate Step Size Determination

11.3.1 The Basic Idea

In Chapter 10, the constrained steepest descent (CSD) algorithm was presented. There it was proposed to calculate the step size using the golden section method. Although that method is quite good among the interval reducing methods, it may be inappropriate for many engineering applications. The method can require too many function evaluations, which for many engineering problems require solution of a complex analysis problem. Therefore in most practical implementations of algorithms, an *inaccurate line search* that has worked fairly well is

used to determine an approximate step size. We shall describe the procedure and illustrate its use in an example.

The philosophy of the inaccurate line search that we shall present is quite similar to the Armijo's procedure that was presented for unconstrained problems in Chapter 9. There the cost function was used to determine the approximate step size; here the descent function $\Phi^{(k)} = f^{(k)} + RV^{(k)}$ defined in Eq. (10.28) will be used. The basic idea of the approach is to try different step sizes until the condition of sufficient reduction in the descent function is satisfied. To determine an acceptable step size, define a sequence of trial step sizes t_j as follows:

$$t_j = \left(\frac{1}{2}\right)^j; \quad j = 0, 1, 2, 3, 4, \dots \quad (11.25)$$

Thus an acceptable step size will be one of the numbers in the sequence $\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \dots\}$ of trial step sizes. Basically, we start with the trial step size as $t_0 = 1$. If a certain descent condition (defined in the following paragraph) is not satisfied, the trial step is taken as half of the previous trial, i.e., $t_1 = \frac{1}{2}$. If the descent condition is still not satisfied, the trial step size is bisected again. The procedure is continued, until the descent condition is satisfied.

11.3.2 Descent Condition

In the following development, we shall use a *second subscript* or *superscript* to indicate values of certain quantities at the trial step sizes. For example, let t_j be the trial step size at the k th iteration. Then the trial design point for which the descent condition is checked is calculated as

$$\mathbf{x}^{(k+1,j)} = \mathbf{x}^{(k)} + t_j \mathbf{d}^{(k)} \quad (11.26)$$

At the k th iteration, we determine an acceptable step size as $\alpha_k = t_j$, with j as the smallest integer (or, largest number in the sequence $1, \frac{1}{2}, \frac{1}{4}, \dots$) to satisfy the *descent condition*

$$\Phi_{k+1,j} \leq \Phi_k - t_j \beta_k \quad (11.27)$$

where $\Phi_{k+1,j}$ is the descent function of Eq. (10.28) evaluated at the trial step size t_j and the corresponding design point $\mathbf{x}^{(k+1,j)}$ as

$$\Phi_{k+1,j} = \Phi(\mathbf{x}^{(k+1,j)}) = f_{k+1,j} + RV_{k+1,j} \quad (11.28)$$

with $f_{k+1,j} = f(\mathbf{x}^{(k+1,j)})$ and $V_{k+1,j} \geq 0$ as the maximum constraint violation at the trial design point calculated using Eq. (10.32). Note that in evaluating $\Phi_{k+1,j}$ and Φ_k in Eq. (11.27), the most recent value of the penalty parameter R is used. The constant β_k in Eq. (11.27) is determined using the search direction $\mathbf{d}^{(k)}$ as

$$\beta_k = \gamma \|\mathbf{d}^{(k)}\|^2 \quad (11.29)$$

where γ is a specified constant between 0 and 1. We shall later study the effect of γ on the step size determination process. Note that in the k th iteration β_k defined in Eq. (11.29) is a constant. As a matter of fact t_j is the only variable on the right side of Inequality (11.27). However, when t_j is changed, the design point is changed, affecting the cost and constraint function values. This way the descent function value on the left side of Inequality (11.27) is changed.

Inequality (11.27) is called the *descent condition*. It is an important condition that must be satisfied at each iteration to obtain a convergent algorithm. To understand the meaning of condition (11.27), consider Fig. 11-1, where various quantities are plotted as functions of t . For example, the horizontal line A-B represents the constant Φ_k , which is the value of the

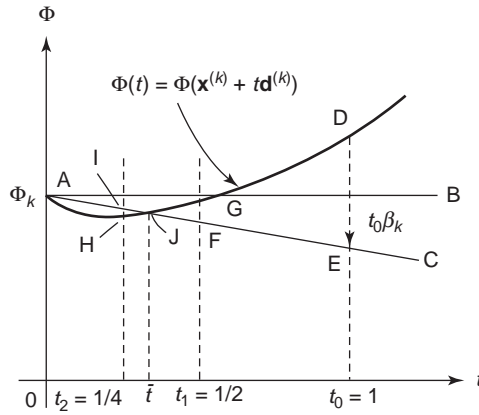


FIGURE 11-1 Geometrical interpretation of the descent condition for determination of step size in the constrained steepest descent algorithm.

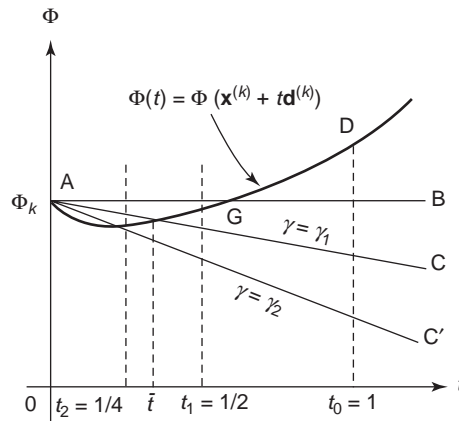


FIGURE 11-2 Effect of parameter γ on step size determination.

descent function at the current design point $\mathbf{x}^{(k)}$; line A–C represents the function whose origin has been moved to point A; and the curve AHGD represents the descent function Φ plotted as a function of the parameter t and originating from point A. The line A–C and the curve AHGD intersect at point J which corresponds to the point $t = \bar{t}$ on the t -axis. For the descent condition of Inequality (11.27) to be satisfied, the curve AHGD must be below the line A–C. This gives only the portion AHJ of the curve AHGD. Thus, we see from the figure that a step size larger than \bar{t} does not satisfy the descent condition of Inequality (11.27). To verify this, consider points D and E on the line $t_0 = 1$. Point D represents $\Phi_{k+1,0} = \Phi(\mathbf{x}^{(k+1,0)})$ and point E represents $(\Phi_k - t_0\beta_k)$. Thus point D represents the left side (LS) and point E represents the right side (RS) of the Inequality (11.27). Since point D is higher than point E, Inequality (11.27) is violated. Similarly, points G and F on the line $t_1 = \frac{1}{2}$ violate the descent condition. Points I and H on the line $t_2 = \frac{1}{4}$ satisfy the descent condition, so the step size α_k at the k th iteration is taken as $\frac{1}{4}$ for the example of Fig. 11-1.

It is important to understand the effect of γ on step size determination. γ is selected as a positive number between 0 and 1. Let us select γ_1 and $\gamma_2 > 0$ with $\gamma_2 > \gamma_1$. Larger γ gives a larger value for the constant β_k in Eq. (11.29). Since β_k is the slope of the line $t\beta_k$, we designate line A–C as $\gamma = \gamma_1$ and A–C' as $\gamma = \gamma_2$ in Fig. 11-2. Thus, we observe from the figure that a larger γ tends to reduce the step size in order to satisfy the descent condition of Inequal-

ity (11.27). For the purpose of checking the descent condition in actual calculations, it may be more convenient to write the inequality of Eq. (11.27) as

$$\Phi_{k+1,j} + t_j \beta_k \leq \Phi_k; \quad j = 0, 1, 2 \dots \quad (11.30)$$

We illustrate the procedure for calculating step size in Example 11.4.

EXAMPLE 11.4 Calculations for Step Size in Constrained Steepest Descent Method

An engineering design problem is formulated as

$$\text{minimize } f(\mathbf{x}) = x_1^2 + 320x_1x_2 \quad (a)$$

subject to the constraints

$$g_1(\mathbf{x}) = \frac{x_1}{60x_2} - 1 \leq 0, \quad g_2(\mathbf{x}) = 1 - \frac{x_1(x_1 - x_2)}{3600} \leq 0 \quad (b)$$

$$g_3(\mathbf{x}) = -x_1 \leq 0, \quad g_4(\mathbf{x}) = -x_2 \leq 0 \quad (c)$$

At a design point $\mathbf{x}^{(0)} = (40, 0.5)$, the search direction is given as $\mathbf{d}^{(0)} = (25.6, 0.45)$. The Lagrange multiplier vector for the constraint is given as $\mathbf{u} = [4880 \ 19,400 \ 0 \ 0]^T$. Choose $\gamma = 0.5$ and calculate the step size for design change using the inexact line search procedure.

Solution. Since the Lagrange multipliers for the constraints are given, the initial value of the penalty perimeter is calculated as

$$R = \sum_{i=1}^4 u_i = 4880 + 19,400 = 24,280 \quad (d)$$

It is important to note that same value of R is to be used on both sides of the descent condition in Eq. (11.27) or Eq. (11.30). The constant β_0 of Eq. (11.29) is calculated as

$$\beta_0 = 0.5(25.6^2 + 0.45^2) = 328 \quad (e)$$

Calculation of Φ_0 . The cost and constraint functions at the starting point $\mathbf{x}^{(0)} = (40, 0.5)$ are calculated as

$$f_0 = f(40, 0.5) = 40^2 + 320(40)(0.5) = 8800$$

$$g_1(40, 0.5) = \frac{40}{60(0.5)} - 1 = 0.333 \text{ (violation)}$$

$$g_2(40, 0.5) = 1 - \frac{40(40 - 0.5)}{3600} = 0.5611 \text{ (violation)} \quad (f)$$

$$g_3(40, 0.5) = -40 < 0 \text{ (inactive)}$$

$$g_4(40, 0.5) = -0.5 < 0 \text{ (inactive)}$$

Maximum constraint violation using Eq. (10.32) is given as

$$V_0 = \max\{0; 0.333, 0.5611, -40, -0.5\} = 0.5611 \quad (g)$$

Using Eq. (10.28), the current descent function is evaluated as

$$\Phi_0 = f_0 + RV_0 = 8000 + (24,280)(0.5611) = 21,624 \quad (h)$$

Trial step size $t_0 = 1$. Let $j = 0$ in Eq. (11.25), so the trial step size is $t_0 = 1$. The trial design point is calculated from Eq. (11.26) as

$$\begin{aligned} x_1^{(1,0)} &= x_1^{(0)} + t_0 d_1^{(0)} = 40 + (1.0)(25.6) = 65.6 \\ x_2^{(1,0)} &= x_2^{(0)} + t_0 d_2^{(0)} = 0.5 + (1.0)(0.45) = 0.95 \end{aligned} \quad (i)$$

The cost and constraint functions at the trial design point are calculated as

$$\begin{aligned} f_{1,0} &= f(65.6, 0.95) = (65.6)^2 + 320(65.6)(0.95) = 24,246 \\ g_1(65.6, 0.95) &= \frac{65.6}{60(0.95)} - 1 = 0.151 > 0 \text{ (violation)} \end{aligned} \quad (j)$$

$$\begin{aligned} g_2(65.6, 0.95) &= 1 - \frac{65.6(65.6 - 0.95)}{3600} = -0.1781 < 0 \text{ (inactive)} \\ g_3(65.6, 0.95) &= -65.6 < 0 \text{ (inactive)} \\ g_4(65.6, 0.95) &= -0.95 < 0 \text{ (inactive)} \end{aligned} \quad (k)$$

The maximum constraint violation using Eq. (10.32) is given as

$$V_{1,0} = \max\{0; 0.151, -0.1781, -65.6, -0.95\} = 0.151 \quad (l)$$

The descent function at the first trial point is calculated using Eq. (11.28) as

$$\Phi_{1,0} = f_{1,0} + RV_{1,0} = 24,246 + 24,480(0.151) = 27,912 \quad (m)$$

For the descent condition of Eq. (11.30), we get $LS = 27,912 + 328 = 28,240$ and $RS = 21,624$. Since $LS > RS$, Inequality (11.30) is violated.

Trial step size $t_1 = 0.5$. Let $j = 1$ in Eq. (11.25), so the trial step size $t_1 = 0.5$. The new trial design point is

$$\begin{aligned} x_1^{(1,1)} &= x_1^{(0)} + t_1 d_1^{(0)} = 40 + 0.5(25.6) = 52.8 \\ x_2^{(1,1)} &= x_2^{(0)} + t_1 d_2^{(0)} = 0.5 + 0.5(0.45) = 0.725 \end{aligned} \quad (n)$$

The cost and constraint functions at the new trial design point are calculated as

$$\begin{aligned} f_{1,1} &= f(52.8, 0.725) = (52.8)^2 + 320(52.8)(0.725) = 15,037 \\ g_1(52.8, 0.725) &= \frac{52.8}{60(0.725)} - 1 = 0.2138 > 0 \text{ (violation)} \end{aligned} \quad (o)$$

$$\begin{aligned}
g_2(52.8, 0.725) &= 1 - \frac{52.8(52.8 - 0.725)}{3600} = 0.2362 > 0 \text{ (violation)} \\
g_3(52.8, 0.725) &= -52.8 < 0 \text{ (inactive)} \\
g_4(52.8, 0.725) &= -0.725 < 0 \text{ (inactive)}
\end{aligned} \tag{p}$$

Maximum constraint violation using Eq. (10.32) is given as

$$V_{1,1} = \max\{0; 0.2138, 0.2362, -52.8, -0.725\} = 0.2362 \tag{q}$$

The descent function at the trial design point is calculated using Eq. (10.27) as

$$\Phi_{1,1} = f_{1,1} + RV_{1,1} = 15,037 + 24,280(0.2362) = 20,772 \tag{r}$$

Evaluating the descent condition of Eq. (11.30), we get the left side as $LS = 20,772 + (0.5)(328) = 20,936$, and the right side as $RS = 21,624$. Since $LS < RS$, Inequality (11.30) is satisfied. Therefore the step size of 0.5 is acceptable.

11.3.3 CSD Algorithm with Approximate Step Size

Example 11.5 illustrates calculation of the approximate step size in the CSD algorithm.

EXAMPLE 11.5 Use of Constrained Steepest Descent Algorithm

Consider the problem of Example 10.2:

$$\text{minimize } f(\mathbf{x}) = x_1^2 + x_2^2 - 3x_1x_2 \tag{a}$$

subject to

$$g_1(\mathbf{x}) = \frac{1}{6}x_1^2 + \frac{1}{6}x_2^2 - 1.0 \leq 0, \quad g_2(\mathbf{x}) = -x_1 \leq 0, \quad g_3(\mathbf{x}) = -x_2 \leq 0 \tag{b}$$

Let $\mathbf{x}^{(0)} = (1, 1)$ be the starting design. Use $R_0 = 10$, $\gamma = 0.5$, and $\varepsilon_1 = \varepsilon_2 = 0.001$ in the constrained steepest descent method. Perform only two iterations.

Solution. The functions of the problem are plotted in Fig. 10-4. The optimum solution for the problem is obtained as $\mathbf{x} = (\sqrt{3}, \sqrt{3})$, $\mathbf{u} = (3, 0, 0)$, $f = -3$.

Iteration 1 ($k = 0$). For the CSD method the following steps are implemented.

Step 1. The initial data are specified as $\mathbf{x}^{(0)} = (1, 1)$; $R_0 = 10$; $\gamma = 0.5$ ($0 < \gamma < 1$); $\varepsilon_1 = \varepsilon_2 = 0.001$.

Step 2. To form the QP subproblem, the cost and constraint function values and their gradients must be evaluated at the initial design point $\mathbf{x}^{(0)}$:

$$\begin{aligned}
f(1, 1) &= -1, & \nabla f(1, 1) &= (-1, -1) \\
g_1(1, 1) &= -\frac{2}{3} < 0 \text{ (inactive)} & \nabla g_1(1, 1) &= \left(\frac{1}{3}, \frac{1}{3}\right) \\
g_2(1, 1) &= -1 < 0 \text{ (inactive)} & \nabla g_2(1, 1) &= (-1, 0) \\
g_3(1, 1) &= -1 < 0 \text{ (inactive)} & \nabla g_3(1, 1) &= (0, -1)
\end{aligned} \tag{c}$$

Note that all constraints are inactive at the starting point so $V_0 = 0$ calculated from Eq. (10.32) as $V_0 = \max\{0; -\frac{2}{3}, -1, -1\}$. The linearized constraints are plotted in Fig. 10.5.

Step 3. Using the preceding values, the QP subproblem of Eqs. (10.25) and (10.26) at $(1, 1)$ is given as: minimize $\bar{f} = (-d_1 - d_2) + 0.5(d_1^2 + d_2^2)$ subject to $\frac{1}{3}d_1 + \frac{1}{3}d_2 \leq \frac{2}{3}$, $-d_1 \leq 1$, $-d_2 \leq 1$. Note that the QP subproblem is strictly convex and thus has a unique solution. A numerical method must generally be used to solve the subproblem. However, since the present problem is quite simple, it can be solved by writing the KKT necessary conditions of Theorem 4.6 as follows:

$$\begin{aligned}
L &= (-d_1 - d_2) + 0.5(d_1^2 + d_2^2) + u_1 \left[\frac{1}{3}(d_1 + d_2 - 2) + s_1^2 \right] + u_2(-d_1 - 1 + s_2^2) \\
&\quad + u_3(-d_2 - 1 + s_3^2)
\end{aligned} \tag{d}$$

$$\begin{aligned}
\frac{\partial L}{\partial d_1} &= -1 + d_1 + \frac{1}{3}u_1 - u_2 = 0 \\
\frac{\partial L}{\partial d_2} &= -1 + d_2 + \frac{1}{3}u_1 - u_3 = 0
\end{aligned} \tag{e}$$

$$\begin{aligned}
\frac{1}{3}(d_1 + d_2 - 2) + s_1^2 &= 0 \\
(-d_1 - 1) + s_2^2 &= 0, \quad (-d_2 - 1) + s_3^2 = 0
\end{aligned} \tag{f}$$

$$u_i s_i = 0; \quad \text{and} \quad s_i^2, u_i \geq 0; \quad i = 1, 2, 3 \tag{g}$$

where u_1 , u_2 , and u_3 are the Lagrange multipliers for the three constraints and s_1^2 , s_2^2 , and s_3^2 are the corresponding slack variables. Solving the foregoing KKT conditions, we get the direction vector $\mathbf{d}^{(0)} = (1, 1)$ with $\bar{f} = -1$ and $\mathbf{u}^{(0)} = (0, 0, 0)$. This solution agrees with the graphical solution given in Fig. 10-13. The feasible region for the subproblem is the triangle ABC, and the optimum solution is at Point D.

Step 4. Because $\|\mathbf{d}^{(0)}\| = \sqrt{2} > \varepsilon_2$, the convergence criterion is not satisfied.

Step 5. Calculate $r_0 = \sum_{i=1}^m u_i^{(0)} = 0$ as defined in Eq. (10.31). To satisfy the necessary condition of Inequality (10.30), let $R = \max\{R_0, r_0\} = \max\{10, 0\} = 10$. It is important to note that $R = 10$ is to be used throughout the first iteration to satisfy the descent condition of Eq. (11.27) or Eq. (11.30).

Step 6. For step size determination, we use the inaccurate line search described earlier in this section. The current value of the descent function Φ_0 of Eq. (10.28) and the constant β_0 of Eq. (11.29) are calculated as

$$\Phi_0 = f_0 + R V_0 = -1 + (10)(0) = -1 \tag{h}$$

$$\beta_0 = \gamma \|\mathbf{d}^{(0)}\|^2 = 0.5(1+1) = 1 \tag{i}$$

Let the trial step size be $t_0 = 1$ and evaluate the new value of the descent function to check the descent condition of Eq. (11.27):

$$\mathbf{x}^{(1,0)} = \mathbf{x}^{(0)} + t_0 \mathbf{d}^{(0)} = (2, 2) \quad (\text{j})$$

At the trial design point, evaluate the cost and constraint functions, and then evaluate the maximum constraint violation to calculate the descent function:

$$\begin{aligned} f_{1,0} &= f(2, 2) = -4 \\ V_{1,0} &= V(2, 2) = \max\left\{0; \frac{1}{3} - 2, -2\right\} = \frac{1}{3} \\ \Phi_{1,0} &= f_{1,0} + RV_{1,0} = -4 + (10)\frac{1}{3} = -\frac{2}{3} \\ \Phi_0 - t_0 \beta_0 &= -1 - 1 = -2 \end{aligned} \quad (\text{k})$$

Since $\Phi_{1,0} > \Phi_0 - t_0 \beta_0$, the descent condition of Inequality (11.27) is not satisfied. Let us try $j = 1$ (i.e., bisect the step size to $t_1 = 0.5$), and evaluate the new value of the descent function to check the descent condition of Eq. (11.27). The design is updated as

$$\mathbf{x}^{(1,1)} = \mathbf{x}^{(0)} + t_1 \mathbf{d}^{(0)} = (1.5, 1.5) \quad (\text{l})$$

At the new trial design point, evaluate the cost and constraint functions, and then evaluate the maximum constraint violation to calculate the descent function:

$$\begin{aligned} f_{1,1} &= f(1.5, 1.5) = -2.25 \\ V_{1,1} &= V(1.5, 1.5) = \max\left\{0; -\frac{1}{4}, -1.5, -1.5\right\} = 0 \\ \Phi_{1,1} &= f_{1,1} + RV_{1,1} = -2.25 + (10)0 = -2.25 \\ \Phi_0 - t_1 \beta_0 &= -1 - 0.5 = -1.5 \end{aligned} \quad (\text{m})$$

Now the descent condition of Inequality (11.27) is satisfied (i.e., $\Phi_{1,1} < \Phi_0 - t_1 \beta_0$); thus $\alpha_0 = 0.5$ is acceptable and $\mathbf{x}^{(1)} = (1.5, 1.5)$

Step 7. Set $R_{0+1} = R_0 = 10$, $k = 1$ and go to Step 2.

Iteration 2 ($k = 1$). For the second iteration, Steps 3 to 7 of the CSD algorithm are repeated as follows:

Step 3. The QP subproblem of Eqs. (10.25) and (10.26) at $\mathbf{x}^{(1)} = (1.5, 1.5)$ is defined as follows:

$$\begin{aligned} \text{minimize } \bar{f} &= (-1.5d_1 - 1.5d_2) + 0.5(d_1^2 + d_2^2) \\ \text{subject to } &0.5d_1 + 0.5d_2 \leq 0.25 \text{ and } -d_1 \leq 1.5, -d_2 \leq 1.5 \end{aligned} \quad (\text{n})$$

Since all constraints are inactive, the maximum violation $V_1 = 0$ from Eq. (10.32). The new cost function is given as $f_1 = -2.25$. The solution of the preceding QP subproblem is $\mathbf{d}^{(1)} = (0.25, 0.25)$ and $\mathbf{u}^{(1)} = (2.5, 0, 0)$.

Step 4. As $\|\mathbf{d}^{(1)}\| = 0.3535 > \varepsilon_2$ the convergence criterion is not satisfied.

Step 5. Evaluate $r_1 = \sum_{i=1}^m u_i^{(1)} = 2.5$. Therefore, $R = \max \{R_1, r_1\} = \max \{10, 2.5\} = 10$.

Step 6. For line search, try $j = 0$ in Inequality (11.27) (i.e., $t_0 = 1$):

$$\begin{aligned}\Phi_1 &= f_1 + RV_1 = -2.25 + (10)0 = -2.25 \\ \beta_1 &= \gamma \|\mathbf{d}^{(1)}\|^2 = 0.5(0.125) = 0.0625\end{aligned}\quad (\text{o})$$

Let the trial step size be $t_0 = 1$ and evaluate the new value of the descent function to check the descent condition of Eq. (11.27):

$$\begin{aligned}\mathbf{x}^{(2,0)} &= \mathbf{x}^{(1)} + t_0 \mathbf{d}^{(1)} = (1.75, 1.75) \\ f_{2,0} &= f(1.75, 1.75) = -3.0625\end{aligned}\quad (\text{p})$$

$$\begin{aligned}V_{2,0} &= V(1.75, 1.75) = \max\{0; 0.0208, -1.75, -1.75\} = 0.0208 \\ \Phi_{2,0} &= f_{2,0} + RV_{2,0} = -3.0625 + (10)0.0208 = -2.8541 \\ \Phi_1 - t_0 \beta_1 &= -2.25 - (1)(0.0625) = -2.3125\end{aligned}\quad (\text{q})$$

Because the descent condition of Inequality (11.27) is satisfied, $\alpha_1 = 1.0$ is acceptable and $\mathbf{x}^{(2)} = (1.75, 1.75)$

Step 7. Set $R_2 = R = 10$, $k = 2$ and go to Step 2.

The maximum constraint violation at the new design $\mathbf{x}^{(2)} = (1.75, 1.75)$ is 0.0208, which is greater than the permissible constraint violation. Therefore, we need to go through more iterations to reach the optimum point back to the feasible region. Note, however, that since the optimum point is $(1.732, 1.732)$, the current point is quite close to the solution with $f_2 = -3.0625$. Also, it is observed that the algorithm iterates through the infeasible region for the present problem.

Example 11.6 examines the effect of γ [for use in Eq. (11.29)] on the step size determination in the CSD method.

EXAMPLE 11.6 Effect of γ on the Performance of CSD Algorithm

For the optimum design problem of Example 11.5, study the effect of variations in the parameter γ on the performance of the CSD algorithm.

Solution. In Example 11.5, $\gamma = 0.5$ is used. Let us see what happens if a very small value of γ (say 0.01) is used. All calculations up to Step 6 of Iteration 1 are unchanged. In Step 6, the value of β_0 is changed to $\beta_0 = \gamma \|\mathbf{d}^{(0)}\|^2 = 0.01(2) = 0.02$. Therefore,

$$\Phi_0 - t_0 \beta_0 = -1 - 1(0.02) = -1.02 \quad (\text{a})$$

which is smaller than $\Phi_{1,0}$, so the descent condition of Inequality (11.27) is violated. Thus, the step size in Iteration 1 will be 0.5 as before. Calculations in Iteration 2 are unchanged until Step 6 where $\beta_1 = \gamma \|\mathbf{d}^{(1)}\|^2 = 0.01(0.125) = 0.00125$. Therefore,

$$\Phi_1 - t_0 \beta_1 = -2.25 - (1)(0.00125) = -2.25125 \quad (\text{b})$$

The descent condition of Inequality (11.27) is satisfied. Thus a smaller value of γ has no effect on the first two iterations.

Let us see what happens if a larger value for γ (say 0.9) is chosen. It can be verified that in Iteration 1, there is no difference in calculations. In Step 2, the step size is reduced to 0.5. Therefore, the new design point is $\mathbf{x}^{(2)} = (1.625, 1.625)$. At this point $f_2 = -2.641$, $g_1 = -0.1198$, and $V_1 = 0$. Thus, a larger γ results in a smaller step size and the new design point remains strictly feasible.

Example 11.7 examines the effect of initial value of the penalty parameter R on step size calculation in the CSD method.

EXAMPLE 11.7 Effect of Penalty Parameter R on CSD Algorithm

For the optimum design problem of Example 11.5, study the effect of variations in the parameter R on the performance of the CSD algorithm.

Solution. In Example 11.5, the initial R is selected as 10. Let us see what happens if R is selected as 1.0. There is no change in the calculations up to Step 5 in Iteration 1. In Step 6,

$$\begin{aligned}\Phi_{1,0} &= -4 + (1)\left(\frac{1}{3}\right) = -\frac{11}{3} \\ \Phi_0 - t_0\beta_0 &= -1 + (1)(0) = -1\end{aligned}\quad (a)$$

Therefore, $\alpha_0 = 1$ satisfies the descent condition of Inequality (11.27) and the new design is given as $\mathbf{x}^{(1)} = (2, 2)$. This is different from what was obtained in Example 11.5.

Iteration 2. Since the acceptable step size in Iteration 1 has changed compared with that in Example 11.5, calculations for Iteration 2 need to be performed again.

Step 3. The QP subproblem of Eqs. (10.25) and (10.26) at $\mathbf{x}^{(1)} = (2, 2)$ is defined as follows:

$$\text{minimize } \bar{f} = (-2d_1 - 2d_2) + 0.5(d_1^2 + d_2^2) \quad (b)$$

subject to

$$\frac{2}{3}d_1 + \frac{2}{3}d_2 \leq -\frac{1}{3}, -d_1 \leq 2, -d_2 \leq 2 \quad (c)$$

At the point $(2, 2)$, $V_1 = \frac{1}{3}$ and $f_1 = -4$. The solution of the QP subproblem is given as

$$\mathbf{d}^{(1)} = (-0.25, -0.25) \text{ and } \mathbf{u}^{(1)} = \left(\frac{27}{8}, 0, 0\right) \quad (d)$$

Step 4. As $\|\mathbf{d}^{(1)}\| = 0.3535 > \varepsilon_2$, the convergence criterion is not satisfied.

Step 5. Evaluate $r_1 = \sum_{i=1}^3 u_i^{(1)} = \frac{27}{8}$. Therefore,

$$R = \max\{R_1, r_1\} = \max\left\{1, \frac{27}{8}\right\} = \frac{27}{8} \quad (e)$$

Step 6. For line search, try $j = 0$ in Inequality (11.27), i.e., $t_0 = 1$:

$$\begin{aligned}\Phi_1 &= f_1 + RV_1 = -4 + \left(\frac{27}{8}\right)\left(\frac{1}{3}\right) = -2.875 \\ \Phi_{2,0} &= f_{2,0} + RV_{2,0} = -3.0625 + \left(\frac{27}{8}\right)(0.0208) = -2.9923 \\ \beta_1 &= \gamma \|\mathbf{d}^{(1)}\|^2 = 0.5(0.125) = 0.0625 \\ \Phi_1 - t_0\beta_1 &= -2.875 - (1)(0.0625) = -2.9375\end{aligned}\quad (f)$$

As the descent condition is satisfied, $\alpha_1 = 1.0$ is acceptable and $\mathbf{x}^{(2)} = (1.75, 1.75)$.

Step 7. Set $R_2 = R_1 = \frac{27}{8}$, $k = 2$ and go to Step 2.

The design at the end of the second iteration is the same as in Example 11.5. This is just a coincidence. We observe that a smaller R gave a larger step size in the first iteration. In general this can change the history of the iterative process.

Example 11.8 illustrates use of the CSD method for an engineering design problem.

EXAMPLE 11.8 Minimum Area Design of a Rectangular Beam

For the minimum area beam design problem of Section 3.8, find the optimum solution using the CSD algorithm starting from the points (50, 200) mm and (1000, 1000) mm.

Solution. The problem is formulated and solved graphically in Section 3.8. After normalizing the constraints, we define the problem as follows: find width b and depth d to minimize the cross-sectional area $f(b, d) = bd$ subject to

$$\text{Bending stress constraint: } \frac{(2.40\text{E}+07)}{bd^2} - 1.0 \leq 0 \quad (a)$$

$$\text{Shear stress constraint: } \frac{(1.125\text{E}+05)}{bd} - 1.0 \leq 0 \quad (b)$$

$$\text{Depth constraint: } \frac{d}{2b} - 1.0 \leq 0 \quad (c)$$

$$\text{Explicit bound constraint: } 10 \leq b \leq 1000, \quad 10 \leq d \leq 1000 \quad (d)$$

The graphical solution for the problem is given in Fig. 11-3; any point on the curve AB gives an optimum solution. The problem is solved starting from the given points with the CSD algorithm available in the IDESIGN software package (Arora and Tseng, 1987a,b). The algorithm has been implemented using the potential constraint strategy. Initial data to the program is provided, and it is allowed to run without interruption until convergence is obtained. The constraint violation tolerance and the convergence parameter are set as 0.0001. Iteration histories with the two starting points I and II are shown in Fig. 11-3. Results of the optimization process are summarized in Table

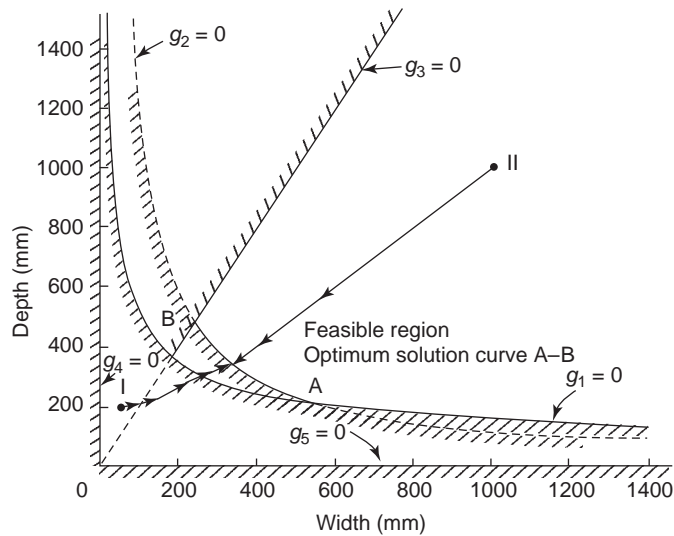


FIGURE 11-3 The history of the iterative process for the rectangular beam design problem.

TABLE 11-2 Results of Optimum Design Process for Rectangular Beam Design Problem

	Starting point I (50, 200)mm	Starting point II (1000, 1000)mm
Optimum point	(315.2, 356.9)	(335.4, 335.4)
Optimum area	1.125×10^5	1.125×10^5
No. of iterations to reach optimum	8	6
No. of calls for function evaluations	8	12
Total no. of constraint gradients evaluated	14	3
Active constraints at optimum	Shear stress	Shear stress
Lagrange multipliers for constraints	1.125×10^5	1.125×10^5

11-2. The starting point I is infeasible with a maximum constraint violation of 1100 percent. The program finds the optimum solution in eight iterations. The algorithm iterates through the infeasible region to reach the optimum solution, which agrees with the one obtained analytically in Section 3.8. The starting point II is feasible and takes six iterations to converge to the optimum. Although the first starting point takes more iterations (eight) to converge to the optimum point compared with the second point (six), the number of calls for function evaluations is smaller for the first point. The total numbers of constraint gradient evaluations with the two points are 14 and 3, respectively. Note that if the potential constraint strategy had not been used, the total number of gradient evaluations would have been 24 and 18, respectively, for the two points. These are substantially higher than the actual number of gradient evaluations with the potential set strategy. It is clear that for large-scale applications, the potential set strategy can have a substantial impact on the efficiency of calculations for an optimization algorithm.

11.4 Constrained Quasi-Newton Methods

Thus far we have used only linear approximation for the cost and constraint functions in defining the search direction determination subproblem. The rate of convergence of algorithms based on such subproblems can be slow. This rate can be improved if second-order information about the problem functions is incorporated into the solution process. It turns out that the QP subproblem defined in Section 10.4 can be modified slightly to introduce curvature information for the Lagrange function into the quadratic cost function of Eq. (10.25) (Wilson, 1963). Since second-order derivatives of the Lagrange function are quite tedious and difficult to calculate, they are approximated using only the first-order information (Han, 1976, 1977; Powell, 1978a,b). The basic idea is the same as for the unconstrained quasi-Newton methods described in Section 9.5. Therefore it is important to review that material at this point. There we used gradients of the cost function at two points for generating the approximate Hessian of the cost function. Instead here, we use the gradient of the Lagrange function at the two points to update approximation to the Hessian of the Lagrange function. These are generally called constrained quasi-Newton methods. They have been also called constrained variable metric (CVM), *sequential quadratic programming* (SQP), or recursive quadratic programming (RQP) methods in the literature. Several variations of the methods can be generated. However, we shall extend the constrained steepest descent algorithm to include the Hessian of the Lagrange function in the definition of the QP subproblem. Derivation of the subproblem is given, and the procedure for updating the approximate Hessian is explained. The idea of constrained quasi-Newton methods is quite simple and straightforward, but very effective in their numerical performance. The method is illustrated with an example and numerical aspects are discussed. The method is used in subsequent chapters to solve several design problems.

11.4.1 Derivation of Quadratic Programming Subproblem

There are several ways to derive the quadratic programming (QP) subproblem that has to be solved at each optimization iteration. Understanding of the detailed derivation of the QP subproblem is not necessary in using the constrained quasi-Newton methods. Therefore, the reader who is not interested in the derivation can skip this subsection. It is customary to derive the QP subproblem by considering only the equality constrained design optimization problem as

$$\begin{aligned} & \text{minimize } f(\mathbf{x}) \\ & \text{subject to } h_i(\mathbf{x}) = 0; \quad i = 1 \text{ to } p \end{aligned} \quad (11.31)$$

Later on, the inequality constraints are easily incorporated into the subproblem. The procedure for the derivation of the QP subproblem is to write KKT necessary conditions of Theorem 4.6 for the problem defined in Eq. (11.31) and then solve the resulting nonlinear equations by Newton's method. Each iteration of Newton's method can be then interpreted as being equivalent to the solution of a QP subproblem. In the following derivations, we assume that all functions are twice continuously differentiable, and gradients of all constraints are linearly independent.

The Lagrange function for the design optimization problem defined in Eq. (11.31) is given as

$$L(\mathbf{x}, \mathbf{v}) = f(\mathbf{x}) + \sum_{i=1}^p v_i h_i(\mathbf{x}) = f(\mathbf{x}) + \mathbf{v} \cdot \mathbf{h}(\mathbf{x}) \quad (11.32)$$

where v_i is the Lagrange multiplier for the i th equality constraint $h_i(\mathbf{x}) = 0$. Note that there is no restriction on the sign of v_i . The KKT necessary conditions give

$$\nabla L(\mathbf{x}, \mathbf{v}) = 0, \quad \nabla f(\mathbf{x}) + \sum_{i=1}^p v_i \nabla h_i(\mathbf{x}) = 0 \quad (11.33)$$

$$h_i(\mathbf{x}) = 0; \quad i = 1 \text{ to } p \quad (11.34)$$

Note that Eq. (11.33) actually represents n equations because the dimension of the design variable vector is n . These equations along with the p equality constraints in Eq. (11.34) give $(n + p)$ equations in $(n + p)$ unknowns (n design variables in \mathbf{x} and p Lagrange multipliers in \mathbf{v}). These are nonlinear equations, so the Newton-Raphson method of Appendix C can be used to solve them. Let us write Eqs. (11.33) and (11.34) in a compact notation as

$$\mathbf{F}(\mathbf{y}) = \mathbf{0} \quad (11.35)$$

where \mathbf{F} and \mathbf{y} are identified as

$$\mathbf{F} = \begin{bmatrix} \nabla L \\ \mathbf{h} \end{bmatrix}_{(n+p \times 1)} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \end{bmatrix}_{(n+p \times 1)} \quad (11.36)$$

Now using the iterative procedure of Section C.2 of Appendix C, we assume that $\mathbf{y}^{(k)}$ at the k th iteration is known and a change $\Delta \mathbf{y}^{(k)}$ is desired. Using linear Taylor's expansion for Eq. (11.35), $\Delta \mathbf{y}^{(k)}$ is given as a solution of the linear system (refer to Eq. C.12 in Appendix C):

$$\nabla \mathbf{F}^T(\mathbf{y}^{(k)}) \Delta \mathbf{y}^{(k)} = -\mathbf{F}(\mathbf{y}^{(k)}) \quad (11.37)$$

where $\nabla \mathbf{F}$ is an $(n + p) \times (n + p)$ Jacobian matrix for the nonlinear equations whose i th column is the gradient of the function $F_i(\mathbf{y})$ with respect to the vector \mathbf{y} . Substituting definitions of \mathbf{F} and \mathbf{y} from Eq. (11.36) into Eq. (11.37), we obtain

$$\begin{bmatrix} \nabla^2 L & \mathbf{N} \\ \mathbf{N}^T & \mathbf{0} \end{bmatrix}^{(k)} \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \mathbf{v} \end{bmatrix}^{(k)} = - \begin{bmatrix} \nabla L \\ \mathbf{h} \end{bmatrix}^{(k)} \quad (11.38)$$

where the superscript k indicates that the quantities are calculated at the k th iteration, $\nabla^2 L$ is an $n \times n$ Hessian matrix of the Lagrange function, \mathbf{N} is an $n \times p$ matrix defined in Eq. (10.23) whose i th column is the gradient of the equality constraint h_i , $\Delta \mathbf{x}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$, and $\Delta \mathbf{v}^{(k)} = \mathbf{v}^{(k+1)} - \mathbf{v}^{(k)}$. Equation (11.38) can be converted to a slightly different form by writing the first row as

$$\nabla^2 L^{(k)} \Delta \mathbf{x}^{(k)} + \mathbf{N}^{(k)} \Delta \mathbf{v}^{(k)} = -\nabla L^{(k)} \quad (11.39)$$

Substituting for $\Delta \mathbf{v}^{(k)} = \mathbf{v}^{(k+1)} - \mathbf{v}^{(k)}$ and ∇L from Eq. (11.33) into Eq. (11.39), we obtain

$$\nabla^2 L^{(k)} \Delta \mathbf{x}^{(k)} + \mathbf{N}^{(k)} (\mathbf{v}^{(k+1)} - \mathbf{v}^{(k)}) = -\nabla f(\mathbf{x}^{(k)}) - \mathbf{N}^{(k)} \mathbf{v}^{(k)} \quad (11.40)$$

Or, the equation is simplified to

$$\nabla^2 L^{(k)} \Delta \mathbf{x}^{(k)} + \mathbf{N}^{(k)} \mathbf{v}^{(k+1)} = -\nabla f(\mathbf{x}^{(k)}) \quad (11.41)$$

Combining Eq. (11.41) with the second row of Eq. (11.38), we obtain

$$\begin{bmatrix} \nabla^2 L & \mathbf{N}^{-\top(k)} \\ \mathbf{N}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x}^{(k)} \\ \mathbf{v}^{(k+1)} \end{bmatrix} = - \begin{bmatrix} \nabla f \\ \mathbf{h} \end{bmatrix} \quad (11.42)$$

Solution of Eq. (11.42) gives a change in the design $\Delta \mathbf{x}^{(k)}$ and a new value for the Lagrange multiplier vector $\mathbf{v}^{(k+1)}$. The foregoing Newton-Raphson iterative procedure to solve KKT necessary conditions is continued until a stopping criterion is satisfied.

It will be now shown that Eq. (11.42) is also the solution of a certain QP problem defined at the k th iteration as (note that the superscript k is omitted for simplicity of presentation):

$$\text{minimize } \nabla f^T \Delta \mathbf{x} + 0.5 \Delta \mathbf{x}^T \nabla^2 L \Delta \mathbf{x} \quad (11.43)$$

subject to linearized equality constraints

$$h_i + \mathbf{n}^{(i)T} \Delta \mathbf{x} = 0; \quad i = 1 \text{ to } p \quad (11.44)$$

where $\mathbf{n}^{(i)}$ is the gradient of the function h_i . The Lagrange function of Eq. (4.46a) for the problem defined in Eqs. (11.43) and (11.44) is given as

$$\bar{L} = \nabla f^T \Delta \mathbf{x} + 0.5 \Delta \mathbf{x}^T \nabla^2 L \Delta \mathbf{x} + \sum_{i=1}^p v_i (h_i + \mathbf{n}^{(i)T} \Delta \mathbf{x}) \quad (11.45)$$

The KKT necessary conditions of Theorem 4.6 treating $\Delta \mathbf{x}$ as the unknown variable give

$$\nabla \bar{L} = \mathbf{0}; \quad \nabla f + \nabla^2 L \Delta \mathbf{x} + \mathbf{N} \mathbf{v} = \mathbf{0} \quad (11.46)$$

$$h_i + \mathbf{n}^{(i)T} \Delta \mathbf{x} = 0; \quad i = 1 \text{ to } p \quad (11.47)$$

It can be seen that if we combine Eqs. (11.46) and (11.47) and write them in a matrix form, we get Eq. (11.42). Thus, the problem of minimizing $f(\mathbf{x})$ subject to $h_i(\mathbf{x}) = 0; i = 1 \text{ to } p$ can be solved by iteratively solving the QP subproblem defined in Eqs. (11.43) and (11.44).

Just as in Newton's method for unconstrained problems, the solution $\Delta \mathbf{x}$ is treated as a search direction and step size is determined by minimizing an appropriate descent function to obtain a convergent algorithm. Defining the search direction \mathbf{d} as $\Delta \mathbf{x}$ and including inequality constraints, the QP subproblem for the general constrained optimization problem is defined as

$$\text{minimize } \bar{f} = \mathbf{c}^T \mathbf{d} + 0.5 \mathbf{d}^T \mathbf{H} \mathbf{d} \quad (11.48)$$

subject to constraints of Eqs. (11.3) and (11.4) as

$$\mathbf{n}^{(i)T} \mathbf{d} = e_i; \quad i = 1 \text{ to } p \quad (11.49)$$

$$\mathbf{a}^{(i)T} \mathbf{d} \leq b_i; \quad i = 1 \text{ to } m \quad (11.50)$$

where the notation defined in Section 10.2 is used, \mathbf{c} is the gradient of the cost function, and \mathbf{H} is the Hessian matrix $\nabla^2 L$ or its approximation. Usually, a potential constraint strategy can be used in reducing the number of inequalities in Eq. (11.50) as discussed in Section 11.1. We shall further elaborate on this point later.

11.4.2 Quasi-Newton Hessian Approximation

Just as for the quasi-Newton methods of Section 9.5 for unconstrained problems, we can approximate the Hessian of the Lagrange function for the constrained problems. We assume that the approximate Hessian $\mathbf{H}^{(k)}$ at the k th iteration is available and we desire to update it to $\mathbf{H}^{(k+1)}$. The BFGS formula of Section 9.5 for direct updating of the Hessian can be used. It is important to note that the updated Hessian should be kept positive definite because, with this property, the QP subproblem defined in Eqs. (11.48) to (11.50) remains strictly convex. Thus a unique search direction is obtained. It turns out that the standard BFGS updating formula can lead to a singular or indefinite Hessian. To overcome this difficulty, Powell (1978a) suggested a modification to the standard BFGS formula. Although the modification is based on intuition, it has worked well in most applications. We shall give the modified BFGS formula.

Several intermediate scalars and vectors must be calculated before the final formula can be given. We define these as follows:

Design change vector ($\alpha_k =$ step size):

$$\mathbf{s}^{(k)} = \alpha_k \mathbf{d}^{(k)} \quad (11.51)$$

Vector:

$$\mathbf{z}^{(k)} = \mathbf{H}^{(k)} \mathbf{s}^{(k)} \quad (11.52)$$

Difference in the gradients of the Lagrange function at two points:

$$\mathbf{y}^{(k)} = \nabla L(\mathbf{x}^{(k+1)}, \mathbf{u}^{(k)}, \mathbf{v}^{(k)}) - \nabla L(\mathbf{x}^{(k)}, \mathbf{u}^{(k)}, \mathbf{v}^{(k)}) \quad (11.53)$$

Scalar:

$$\xi_1 = \mathbf{s}^{(k)} \cdot \mathbf{y}^{(k)} \quad (11.54)$$

Scalar:

$$\xi_2 = \mathbf{s}^{(k)} \cdot \mathbf{z}^{(k)} \quad (11.55)$$

Scalar:

$$\theta = 1 \text{ if } \xi_1 \geq 0.2 \xi_2, \text{ otherwise } \theta = 0.8 \xi_2 / (\xi_2 - \xi_1) \quad (11.56)$$

Scalar:

$$\mathbf{w}^{(k)} = \theta \mathbf{y}^{(k)} + (1 - \theta) \mathbf{z}^{(k)} \quad (11.57)$$

Scalar:

$$\xi_3 = \mathbf{s}^{(k)} \cdot \mathbf{w}^{(k)} \quad (11.58)$$

An $n \times n$ correction matrix:

$$\mathbf{D}^{(k)} = (1/\xi_3) \mathbf{w}^{(k)} \mathbf{w}^{(k)T} \quad (11.59)$$

An $n \times n$ correction matrix:

$$\mathbf{E}^{(k)} = (1/\xi_2) \mathbf{z}^{(k)} \mathbf{z}^{(k)T} \quad (11.60)$$

With the preceding definition of matrices $\mathbf{D}^{(k)}$ and $\mathbf{E}^{(k)}$, the Hessian is updated as

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \mathbf{D}^{(k)} - \mathbf{E}^{(k)} \quad (11.61)$$

It turns out that if the scalar ξ_1 in Eq. (11.54) is negative, the original BFGS formula can lead to an indefinite Hessian. The use of the modified vector $\mathbf{w}^{(k)}$ given in Eq. (11.57) tends to alleviate this difficulty. Because of the usefulness of incorporating a Hessian into an optimization algorithm, several updating procedures have been developed in the literature (Gill *et al.*, 1981). For example, Cholesky factors of the Hessian can be directly updated. In numerical implementations, it is useful to incorporate such procedures because numerical stability can be guaranteed.

11.4.3 Modified Constrained Steepest Descent Algorithm

The CSD algorithm of Section 10.5 has been extended to include Hessian updating and potential set strategy (Belegundu and Arora, 1984a; Lim and Arora, 1986; Thanedar *et al.*, 1986; Huang and Arora, 1996). The original algorithm did not use the potential set strategy (Han, 1976, 1977; Powell, 1978a,b,c). The new algorithm has been extensively investigated numerically and several computational enhancements have been incorporated into it to make it robust as well as efficient. In the following, we describe a very basic algorithm as a simple extension of the CSD algorithm. We refer to the new algorithm that uses a potential set strategy as the *SQP method*:

Step 1. The same as the CSD algorithm of Section 10.5, except also set the initial estimate or the approximate Hessian as identity, i.e. $\mathbf{H}^{(0)} = \mathbf{I}$.

Step 2. Calculate the cost and constraint functions at $\mathbf{x}^{(k)}$ and calculate the gradients of cost and constraint functions. Calculate the maximum constraint violation V_k as defined in Eq. (10.32). If $k > 0$, update the Hessian of the Lagrange function using Eqs. (11.51) to (11.61). If $k = 0$, skip updating and go to Step 3.

Step 3. Define the QP subproblem of Eqs. (11.48) to (11.50) and solve it for the search direction $\mathbf{d}^{(k)}$ and Lagrange multipliers $\mathbf{v}^{(k)}$ and $\mathbf{u}^{(k)}$.

Steps 4–7. Same as for the CSD algorithm of Section 10.5.3.

Thus, we see that the only difference between the two algorithms is in Steps 2 and 3. We demonstrate use of the SQP algorithm with Example 11.9.

EXAMPLE 11.9 Use of SQP Method

Complete two iterations of the SQP algorithm for Example 11.5:

$$\text{minimize } f(\mathbf{x}) = x_1^2 + x_2^2 - 3x_1x_2 \quad (a)$$

subject to

$$g_1(\mathbf{x}) = \frac{1}{6}x_1^2 + \frac{1}{6}x_2^2 - 1.0 \leq 0, \quad g_2(\mathbf{x}) = -x_1 \leq 0, \quad g_3(\mathbf{x}) = -x_2 \leq 0. \quad (b)$$

The starting point is (1, 1), $R_0 = 10$, $\gamma = 0.5$, $\varepsilon_1 = \varepsilon_2 = 0.001$.

Solution. The first iteration of the SQP algorithm is the same as the CSD algorithm. From Example 11.5, results of the first iteration are

$$\begin{aligned} \mathbf{d}^{(0)} &= (1, 1); & \alpha_0 &= 0.5, & \mathbf{x}^{(1)} &= (1.5, 1.5) \\ \mathbf{u}^{(0)} &= (0, 0, 0); & R_1 &= 10, & \mathbf{H}^{(0)} &= \mathbf{I}. \end{aligned} \quad (c)$$

Iteration 2. At the point $\mathbf{x}^{(1)} = (1.5, 1.5)$, the cost and constraint functions and their gradients are evaluated as

$$\begin{aligned} f &= -6.75; & \nabla f &= (-1.5, -1.5) \\ g_1 &= -0.25; & \nabla g_1 &= (0.5, 0.5) \\ g_2 &= -1.5; & \nabla g_2 &= (-1, 0) \\ g_3 &= -1.5; & \nabla g_3 &= (0, -1) \end{aligned} \quad (d)$$

To update the Hessian matrix, we define the vectors in Eqs. (11.51) and (11.52) as

$$\mathbf{s}^{(0)} = \alpha_0 \mathbf{d}^{(0)} = (0.5, 0.5), \quad \mathbf{z}^{(0)} = \mathbf{H}^{(0)} \mathbf{s}^{(0)} = (0.5, 0.5) \quad (e)$$

Since the Lagrange multiplier vector $\mathbf{u}^{(0)} = (0, 0, 0)$, the gradient of the Lagrangian ∇L is simply the gradient of the cost function ∇f . Therefore, vector $\mathbf{y}^{(0)}$ of Eq. (11.53) is calculated as

$$\mathbf{y}^{(0)} = \nabla f(\mathbf{x}^{(1)}) - \nabla f(\mathbf{x}^{(0)}) = (-0.5, -0.5) \quad (f)$$

Also, the scalars in Eqs. (11.54) and (11.55) are calculated as

$$\xi_1 = \mathbf{s}^{(0)} \cdot \mathbf{y}^{(0)} = -0.5, \quad \xi_2 = \mathbf{s}^{(0)} \cdot \mathbf{z}^{(0)} = 0.5 \quad (g)$$

Since $\xi_1 < 0.2\xi_2$, θ in Eq. (11.56) is calculated as $\theta = 0.8(0.5)/(0.5 + 0.5) = 0.4$. The vector $\mathbf{w}^{(0)}$ in Eq. (11.57) is calculated as $\mathbf{w}^{(0)} = 0.4(-0.5, -0.5) + (1 - 0.4)(0.5, 0.5) = (0.1, 0.1)$. The scalar ξ_3 in Eq. (11.58) is calculated as $(0.5, 0.5) \cdot (0.1, 0.1) = 0.1$. The two correction matrices in Eqs. (11.59) and (11.60) are calculated as

$$\mathbf{D}^{(0)} = \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix}; \quad \mathbf{E}^{(0)} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} \quad (h)$$

Finally, from Eq. (11.61), the updated Hessian is given as

$$\mathbf{H}^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{bmatrix} - \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.6 & -0.4 \\ -0.4 & 0.6 \end{bmatrix} \quad (i)$$

Step 3. With the updated Hessian and other data previously calculated, the QP subproblem of Eqs. (11.48) to (11.50) is defined as:

minimize $\bar{f} = -1.5d_1 - 1.5d_2 + \frac{1}{2}(0.6d_1^2 - 0.8d_1d_2 + 0.6d_2^2)$ subject to

$$0.5d_1 + 0.5d_2 \leq 0.25, \quad -d_1 \leq 1.5, \quad -d_2 \leq 1.5 \quad (j)$$

The QP subproblem is strictly convex and, thus, has a unique solution. Using KKT conditions the solution is obtained as

$$\mathbf{d}^{(1)} = (0.25, 0.25), \quad \mathbf{u}^{(1)} = (2.9, 0, 0) \quad (k)$$

This solution is the same as in Example 11.5. Therefore, the rest of the steps have the same calculations. It is seen that in this example, inclusion of the approximate Hessian does not actually change the search direction at the second iteration. In general, it will give different directions and better convergence.

11.4.4 Observations on the Constrained Quasi-Newton Methods

The quasi-Newton methods are considered to be most efficient, reliable and generally applicable. Schittkowski and coworkers (1980, 1981, 1987) have extensively analyzed the methods and evaluated them against several other methods using a set of nonlinear programming test problems. Their conclusion is that quasi-Newton methods are far superior to others. Lim and Arora (1986), Thanedar and coworkers (1986), Thanedar, Arora and coworkers (1987) and Arora and Tseng (1987b) have evaluated the methods for a class of engineering design problems. Gabrielle and Beltracchi (1987) have also discussed several enhancements of Pshenichny's constrained steepest descent (CSD) algorithm including incorporation of quasi-Newton updates of the Hessian of the Lagrangian. In general, these investigations have shown the quasi-Newton methods to be superior. Therefore the methods are recommended for general engineering design applications.

Numerical implementation of an algorithm is an art. Considerable care, judgment, safeguards, and user-friendly features must be designed and incorporated into the software. Numerical calculations must be robustly implemented. Each step of the algorithm must be analyzed and proper numerical procedures developed to implement the intent of the step. The software must be properly evaluated for performance by solving many different problems. Many aspects of numerical implementation of algorithms are discussed by Gill and coworkers (1981). The steps of the *SQP algorithm* have been analyzed (Tseng and Arora, 1988). Various potential constraint strategies have been incorporated and evaluated. Several descent functions have been investigated. Procedures to resolve inconsistencies in the QP subproblem have been developed and evaluated. As a result of these enhancements and evaluations, a very powerful, robust, and general algorithm for engineering design applications has become available. The algorithm is used in subsequent chapters to solve engineering design problems.

11.4.5 Descent Functions

Descent functions play an important role in the constrained quasi-Newton methods, so we shall discuss them briefly. Some of the descent functions are nondifferentiable while others are differentiable. For example, the descent function of Eq. (10.27) is nondifferentiable. Another nondifferentiable descent function has been proposed by Han (1977) and Powell (1978c). We shall denote this as Φ_H and define it as follows at the k th iteration:

$$\Phi_H = f(\mathbf{x}^{(k)}) + \sum_{i=1}^p r_i^{(k)} |h_i| + \sum_{i=1}^m \mu_i^{(k)} \max\{0, g_i\} \quad (11.62)$$

where $r_i^{(k)} \geq |v_i^{(k)}|$ are the penalty parameters for equality constraints and $\mu_i^{(k)} \geq u_i^{(k)}$ are the penalty parameters for inequality constraints. The penalty parameters sometimes become very large so Powell (1978c) suggested a procedure to adjust them as follows:

First iteration:

$$r_i^{(0)} = |v_i^{(0)}|; \quad \mu_i^{(0)} = u_i^{(0)} \quad (11.63)$$

Subsequent iterations:

$$\begin{aligned} r_i^{(k)} &= \max\left\{|v_i^{(k)}|, \frac{1}{2}(r_i^{(k-1)} + |v_i^{(k)}|)\right\} \\ \mu_i^{(k)} &= \max\left\{u_i^{(k)}, \frac{1}{2}(\mu_i^{(k-1)} + u_i^{(k)})\right\} \end{aligned} \quad (11.64)$$

Schittkowski (1981) has suggested using the following augmented Lagrangian function Φ_A as the descent function:

$$\Phi_A = f(\mathbf{x}) + P_1(\mathbf{v}, \mathbf{h}) + P_2(\mathbf{u}, \mathbf{g}) \quad (11.65)$$

$$P_1(\mathbf{v}, \mathbf{h}) = \sum_{i=1}^p \left(v_i h_i + \frac{1}{2} r_i h_i^2 \right) \quad (11.66)$$

$$P_2(\mathbf{u}, \mathbf{g}) = \sum_{i=1}^m \begin{cases} u_i g_i + \frac{1}{2} \mu_i g_i^2, & \text{if } (g_i + u_i/\mu_i) \geq 0 \\ \frac{1}{2} u_i^2 / \mu_i, & \text{otherwise} \end{cases} \quad (11.67)$$

where the penalty parameters r_i and μ_i have been defined previously in Eqs. (11.63) and (11.64). One good feature of Φ_A is that the function and its gradient are continuous.

11.5 Other Numerical Optimization Methods

Many other methods and their variations for constrained optimization have been developed and evaluated in the literature. For more details, Gill and coworkers (1981), Luenberger (1984) and Reklaitis and coworkers (1983) should be consulted. In this section, we shall briefly discuss the basic ideas of three methods—the feasible directions, gradient projection, and generalized reduced gradient—that have been used quite successfully for engineering design problems.

11.5.1 Method of Feasible Directions

The method of feasible directions is one of the earliest primal methods for solving constrained optimization problems. The *basic idea of the method is to move from one feasible point to an improved feasible point*. Thus given a feasible design $\mathbf{x}^{(k)}$, an “improving feasible direction” $\mathbf{d}^{(k)}$ is determined such that for a sufficiently small step size $\alpha > 0$, the following two properties are satisfied:

1. The new design, $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}$ is feasible.
2. The new cost function is smaller than the current one, i.e., $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$.

Once $\mathbf{d}^{(k)}$ is determined, a line search is performed to determine how far to proceed along $\mathbf{d}^{(k)}$. This leads to a new feasible design $\mathbf{x}^{(k+1)}$, and the process is repeated from there.

The method is based on the general algorithm described in Section 10.1.1, where the design change determination is decomposed into search direction and step size determination subproblems. The direction is determined by defining a linearized subproblem at the current feasible point, and step size is determined to reduce the cost function as well as maintain feasibility. Since linear approximations are used, it is difficult to maintain feasibility with respect to the equality constraints. Therefore, the method has been developed and applied mostly to inequality constrained problems. Some procedures have been developed to treat equality constraints in these methods. However, we shall describe the method for problems with only inequality constraints.

Now we define a subproblem that yields improving feasible direction at the current design point. An *improving feasible direction* is defined as the one that reduces the cost function as well as remains strictly feasible for a small step size. Thus it is a direction of descent for the cost function as well as pointing toward the inside of the feasible region. The improving feasible direction \mathbf{d} satisfies the conditions $\mathbf{c}^T \mathbf{d} < 0$ and $\mathbf{a}^{(i)T} \mathbf{d} < 0$ for $i \in I_k$ where I_k is a

potential constraint set at the current point as defined in Eq. (11.1). It can be obtained by minimizing the maximum of $\mathbf{c}^T \mathbf{d}$ and $\mathbf{a}^{(i)T} \mathbf{d}$ for $i \in I_k$. Denoting this maximum by β , the *direction finding subproblem* is defined as: minimize β subject to

$$\mathbf{c}^T \mathbf{d} \leq \beta \quad (11.68)$$

$$\mathbf{a}^{(i)T} \mathbf{d} \leq \beta \quad \text{for } i \in I_k \quad (11.69)$$

$$-1 \leq d_j \leq 1; \quad j = 1 \text{ to } n \quad (11.70)$$

The normalization constraint of Eq. (11.70) has been introduced to obtain a bounded solution. Other forms of normalization constraints can also be used. Let (β, \mathbf{d}) be an optimum solution for the previous problem. If $\beta < 0$, then \mathbf{d} is an improving feasible direction. If $\beta = 0$, then the current design point satisfies the KKT necessary conditions.

There are many different line search algorithms that may be used to determine the appropriate step size along the search direction. Also, to determine a better feasible direction $\mathbf{d}^{(k)}$, the constraints of Eq. (11.69) can be expressed as $\mathbf{a}^{(i)T} \mathbf{d} \leq \theta_i \beta$, where $\theta_i > 0$ are the “push-off” factors. 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. Figure 11-4 shows the physical significance of θ_i in the direction-finding subproblem. It depicts a two-variable design space with one active constraint. If θ_i is taken as zero, then the right-side of Eq. (11.69) ($\theta_i \beta$) becomes zero. The direction \mathbf{d} in this case tends to follow the active constraint, i.e., it is tangent to the constraint surface. On the other hand, if θ_i is very large, the direction \mathbf{d} tends to follow the cost function contour. Thus, a small value of θ_i will result in a direction which rapidly reduces the cost function. It may, however, rapidly encounter the same constraint surface due to nonlinearities. Larger values of θ_i will reduce the risk of re-encountering the same constraint, but will not reduce the cost function as fast. A value of $\theta_i = 1$ yields acceptable results for most problems.

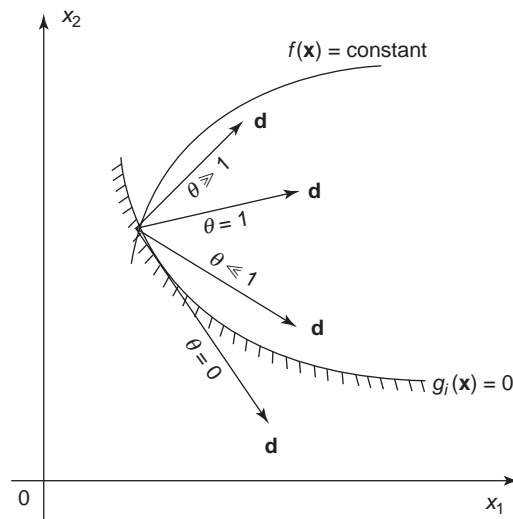


FIGURE 11-4 The effect of push-off factor θ_i on search direction \mathbf{d} in the feasible directions method.

Disadvantages of the method are (1) a feasible starting point is needed—special algorithms must be used to obtain such a point if it is not known—and (2) equality constraints are difficult to impose and require special procedures for their implementation.

11.5.2 Gradient Projection Method

The gradient projection method was developed by Rosen in 1961. Just as in the feasible directions method, the method also uses first-order information about the problem at the current point. The feasible directions method requires the solution of an LP at each iteration to find the search direction. In some applications, this can be an expensive calculation. Thus, Rosen was motivated to develop a method that does not require the solution of an LP. His idea was to develop a procedure in which the direction vector could be calculated easily, although it may not be as good as the one obtained from the feasible directions approach. Thus, he derived an explicit expression for the search direction.

In this method, if the initial point is inside the feasible set, the steepest descent direction for the cost function is used until a constraint boundary is encountered. If the starting point is infeasible, then the constraint correction step is used to reach the feasible set. When the point is on the boundary, a direction that is tangent to the constraint surface is calculated and used to change the design. This direction is computed by projecting the steepest descent direction for the cost function on to the tangent hyperplane. This was termed the constrained steepest descent (CSD) direction in Section 10.5. A step is executed in the negative projected gradient direction. Since the direction is tangent to the constraint surface, the new point will be infeasible. Therefore, a series of correction steps need to be executed to reach the feasible set.

The iterative process of the gradient projection method is illustrated in Fig. 11-5. At the point $\mathbf{x}^{(k)}$, $-\mathbf{c}^{(k)}$ is the steepest descent direction and $\mathbf{d}^{(k)}$ is the negative projected gradient (constrained steepest descent) direction. An arbitrary step takes the point $\mathbf{x}^{(k)}$ to $\mathbf{x}^{(k,1)}$ from where constraint correction steps are executed to reach the feasible point $\mathbf{x}^{(k,1)}$. Comparing the gradient projection method and the constrained steepest descent method of Section 10.5, we observe that at a feasible point where some constraints are active, the two methods have identical directions. The only difference is in the step size determination.

Philosophically, the idea of the gradient projection method is quite good, i.e., the search direction is easily computable, although it may not be as good as the feasible direction. However, numerically the method has considerable uncertainty. The step size specification is arbitrary; the constraint correction process is quite tedious. A serious drawback is that

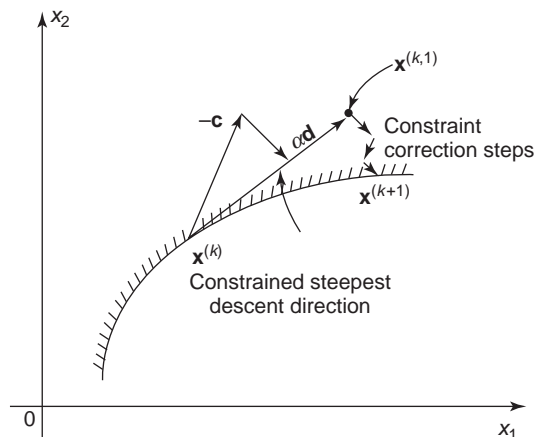


FIGURE 11-5 Steps of the gradient projection method.

convergence of the algorithm is tedious to enforce. For example, during the constraint correction steps, it must be ensured that $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$. If this condition cannot be satisfied or constraints cannot be corrected, then the step size must be reduced and the entire process must be repeated from the previously updated point. This can be tedious to implement, resulting in additional calculations. Despite these drawbacks, the method has been applied quite successfully to a wide variety of engineering design problems (Haug and Arora, 1979). In addition, many variations of the method have been investigated in the literature (Gill *et al.* 1981; Luenberger, 1984; Belegundu and Arora, 1985).

11.5.3 Generalized Reduced Gradient Method

In 1967, Wolfe developed the reduced gradient method based on a simple variable elimination technique for equality constrained problems (Abadie, 1970). The generalized reduced gradient (GRG) method is an extension of the reduced gradient method to accommodate nonlinear inequality constraints. In this method, a search direction is found such that for any small move, the current active constraints remain precisely active. If some active constraints are not precisely satisfied because of nonlinearity of constraint functions, the Newton-Raphson method is used to return to the constraint boundary. Thus, the GRG method can be considered somewhat similar to the gradient projection method.

Since inequality constraints can always be converted to equalities by adding slack variables, we can form an equality constrained NLP model. Also, we can employ the potential constraint strategy and treat all the constraints in the subproblem as equalities. The direction-finding subproblem in the GRG method can be defined in the following way (Abadie and Carpenter, 1969): Let us partition the design variable vector \mathbf{x} as $[\mathbf{y}^T, \mathbf{z}^T]^T$, where $\mathbf{y}_{(n-p)}$ and $\mathbf{z}_{(p)}$ are vectors of independent and dependent design variables, respectively. First-order changes in the cost and constraint functions (treated as equalities) are given as

$$\Delta f = \frac{\partial f^T}{\partial \mathbf{y}} \Delta \mathbf{y} + \frac{\partial f^T}{\partial \mathbf{z}} \Delta \mathbf{z} \quad (11.71)$$

$$\Delta h_i = \frac{\partial h_i^T}{\partial \mathbf{y}} \Delta \mathbf{y} + \frac{\partial h_i^T}{\partial \mathbf{z}} \Delta \mathbf{z} \quad (11.72)$$

Since we started with a feasible design, any change in the variables must keep the current equalities satisfied at least to first order, i.e., $\Delta h_i = 0$. Therefore, using Eq. (11.72) this requirement is written in the matrix form as

$$\mathbf{A}^T \Delta \mathbf{y} + \mathbf{B}^T \Delta \mathbf{z} = \mathbf{0}, \quad \text{or} \quad \Delta \mathbf{z} = -(\mathbf{B}^{-T} \mathbf{A}^T) \Delta \mathbf{y} \quad (11.73)$$

where columns of matrices $\mathbf{A}_{((n-p) \times p)}$ and $\mathbf{B}_{(p \times p)}$ contain gradients of equality constraints with respect to \mathbf{y} and \mathbf{z} , respectively. Equation (11.73) can be viewed as the one that determines $\Delta \mathbf{z}$ (change in the dependent variable) when $\Delta \mathbf{y}$ (change in the independent variable) is specified. Substituting $\Delta \mathbf{z}$ from Eq. (11.73) into Eq. (11.71) we can calculate Δf and identify $df/d\mathbf{y}$ as

$$\Delta f = \left(\frac{\partial f^T}{\partial \mathbf{y}} - \frac{\partial f^T}{\partial \mathbf{z}} \mathbf{B}^{-T} \mathbf{A}^T \right) \Delta \mathbf{y}; \quad \frac{df}{d\mathbf{y}} = \frac{\partial f}{\partial \mathbf{y}} - \mathbf{A} \mathbf{B}^{-1} \frac{\partial f}{\partial \mathbf{z}} \quad (11.74)$$

$df/d\mathbf{y}$ is commonly known as the reduced gradient. In line search, the cost function is treated as the descent function. For a trial value of α , the design variables are updated using $\Delta \mathbf{y} = -\alpha df/d\mathbf{y}$ and $\Delta \mathbf{z}$ from Eq. (11.73). If the trial design is not feasible, then independent

design variables are considered to be fixed and dependent variables are changed iteratively by applying the Newton-Raphson method [Eq. (11.73)] until we get a feasible design point. If the new feasible design satisfies the descent condition, then line search is terminated; otherwise, the previous trial step size is discarded and the procedure is repeated with a reduced step size. It can be observed that when $df/dy = \mathbf{0}$ in Eq. (11.74), the KKT conditions of optimality are satisfied for the original NLP problem.

The main computational burden associated with the GRG algorithm arises from the Newton-Raphson iterations during line search. Strictly speaking, the gradients of constraints need to be recalculated and the Jacobian matrix \mathbf{B} needs to be inverted at every iteration during the line search. This is prohibitively expensive. Toward this end, many efficient numerical schemes have been suggested, e.g., the use of a quasi-Newton formula to update \mathbf{B}^{-1} without recomputing gradients but requiring only constraint function values. This can cause problems if the set of independent variables changes during iterations. Another difficulty is to select a feasible starting point. Special algorithms must be used to handle arbitrary starting points, as in the feasible directions method.

There is some confusion in the literature on the relative merits and demerits of the reduced gradient method. For example, the method has been declared superior to the gradient projection method, whereas the two methods are considered essentially the same by Sargeant (1974). The confusion arises when studying the reduced gradient method in the context of solving inequality constrained problems; some algorithms convert the inequalities into equalities by adding nonnegative slack variables while others adopt potential constraint strategy. It turns out that if a potential constraint strategy is used, the reduced gradient method becomes essentially the same as the gradient projection method (Belegundu and Arora, 1985). On the other hand, if inequalities are converted to equalities, it behaves quite differently from the gradient projection method. Unfortunately, the inequality constrained problem in most engineering applications must be solved using a potential constraint strategy, as the addition of slack variables to inequalities implies that all constraints are active at every iteration and must, therefore, be differentiated. This is ruled out for large-scale applications because of the enormous computation and storage of information involved. Therefore, as observed by Belegundu and Arora (1985), we need not differentiate between gradient projection and reduced gradient methods when solving most engineering optimization problems.

Exercises for Chapter 11

Section 11.3 Approximate Step Size Determination

For the following problems, complete one iteration of the constrained steepest descent method for the given starting point (let $R_0 = 1$ and $\gamma = 0.5$, use the approximate step size determination procedure).

- 11.1 Beam design problem formulated in Section 3.8 at the point $(b, d) = (250, 300)$ mm.
- 11.2 Tubular column design problem formulated in Section 2.7 at the point $(R, t) = (12, 4)$ cm. Let $P = 50$ kN, $E = 210$ GPa, $l = 500$ cm, $\sigma_a = 250$ MPa, and $\rho = 7850$ kg/m³.
- 11.3 Wall bracket problem formulated in Section 4.7.1 at the point $(A_1, A_2) = (150, 150)$ cm².
- 11.4 Exercise 2.1 at the point $h = 12$ m, $A = 4000$ m².
- 11.5 Exercise 2.3 at the point $(R, H) = (6, 15)$ cm.

- 11.6 Exercise 2.4 at the point $R = 2$ cm, $N = 100$.
- 11.7 Exercise 2.5 at the point $(W, D) = (100, 100)$ m.
- 11.8 Exercise 2.9 at the point $(r, h) = (6, 16)$ cm.
- 11.9 Exercise 2.10 at the point $(b, h) = (5, 10)$ m.
- 11.10 Exercise 2.11 at the point, width = 5 m, depth = 5 m, and height = 5 m.
- 11.11 Exercise 2.12 at the point $D = 4$ m and $H = 8$ m.
- 11.12 Exercise 2.13 at the point $w = 10$ m, $d = 10$ m, $h = 4$ m.
- 11.13 Exercise 2.14 at the point $P_1 = 2$ and $P_2 = 1$.

Section 11.4 Constrained Quasi-Newton Methods

Complete two iterations of the constrained quasi-Newton method and compare the search directions with the ones obtained with the CSD algorithm (note that the first iteration is the same for both methods; let $R_0 = 1$, $\gamma = 0.5$).

- 11.14 Beam design problem formulated in Section 3.8 at the point $(b, d) = (250, 300)$ mm.
- 11.15 Tubular column design problem formulated in Section 2.7 at the point $(R, t) = (12, 4)$ cm. Let $P = 50$ kN, $E = 210$ GPa, $l = 500$ cm, $\sigma_a = 250$ MPa, and $\rho = 7850$ kg/m³.
- 11.16 Wall bracket problem formulated in Section 4.7.1 at the point $(A_1, A_2) = (150, 150)$ cm².
- 11.17 Exercise 2.1 at the point $h = 12$ m, $A = 4000$ m².
- 11.18 Exercise 2.3 at the point $(R, H) = (6, 15)$ cm.
- 11.19 Exercise 2.4 at the point $R = 2$ cm, $N = 100$.
- 11.20 Exercise 2.5 at the point $(W, D) = (100, 100)$ m.
- 11.21 Exercise 2.9 at the point $(r, h) = (6, 16)$ cm.
- 11.22 Exercise 2.10 at the point $(b, h) = (5, 10)$ m.
- 11.23 Exercise 2.11 at the point, width = 5 m, depth = 5 m, and height = 5 m.
- 11.24 Exercise 2.12 at the point $D = 4$ m and $H = 8$ m.
- 11.25 Exercise 2.13 at the point $w = 10$ m, $d = 10$ m, $h = 4$ m.
- 11.26 Exercise 2.14 at the point $P_1 = 2$ and $P_2 = 1$.

Formulate and solve the following problems using Excel Solver or other software.

- | | | |
|----------------------|----------------------|----------------------|
| 11.27* Exercise 3.34 | 11.28* Exercise 3.35 | 11.29* Exercise 3.36 |
| 11.30* Exercise 3.50 | 11.31* Exercise 3.51 | 11.32* Exercise 3.52 |
| 11.33* Exercise 3.53 | 11.34* Exercise 3.54 | |

12 Introduction to Optimum Design with MATLAB

Upon completion of this chapter you will be able to:

- Use the Optimization Toolbox in MATLAB to solve unconstrained and constrained problems

MATLAB was used in Chapter 3 to graphically solve two variable optimization problems. In Chapter 4 it was used to solve a set of nonlinear equations obtained as KKT optimality conditions for constrained optimization problems. In this chapter, we describe capabilities of the Optimization Toolbox in MATLAB to solve linear, quadratic, and nonlinear programming problems. We start by describing the basic capabilities of this toolbox. Some operators and syntax used to enter expressions and data are described. In subsequent sections, we illustrate the use of the program for unconstrained and constrained optimization problems. Some engineering design optimization problems are also solved using the program.

12.1 Introduction to Optimization Toolbox

12.1.1 Variables and Expressions

MATLAB can be considered a high-level programming language for numerical computation, data analysis, and graphics for applications in many fields. It interprets and evaluates expressions entered at the keyboard. The statements are usually in the form “variable = expression.” The variables can be scalars, arrays, or matrices. Arrays may store many variables at a time. A simple way to define a scalar, array, or matrix is to use *assignment statements* as follows:

```
a = 1;    b = [1, 1];    c = [1, 0, 0; 1, 1, 0; 1, -2, 1];
```

Note that several assignment statements can be entered in one row. A semicolon “;” at the end of a statement prevents the program from executing the statement interactively and displaying the results immediately. The variable *a* denotes a scalar that is assigned a value of 1;

the variable \mathbf{b} denotes a 1×2 row vector and the variable \mathbf{c} denotes a 3×3 matrix assigned as follows:

$$\mathbf{c} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

The semicolons in the brackets of expression for \mathbf{c} separate the rows, and the values in the rows can be separated by commas or blanks. MATLAB has a rule that the *variable name* must be a single word without spaces, and start with a letter followed by any number of letters, digits, or underscores. It is also important to note that the variable names are case sensitive. In addition, there are several built-in variables; e.g., “pi” for the ratio of the circumference of a circle to its diameter, “esp” for the smallest number in the computer, “inf” for infinity, and so on.

12.1.2 Scalar, Array, and Matrix Operations

The arithmetic operators for scalars in MATLAB are: addition “+”, subtraction “-”, multiplication “*”, division “/”, and exponentiation “^”. Vector and matrix calculations can also be organized in a simple way using these operators. For example, multiplication of two matrices \mathbf{A} and \mathbf{B} is expressed as $\mathbf{A} * \mathbf{B}$. Slight modification of the standard operators is used for element-by-element operations between vectors and matrices: “.*” for multiplication, “./” for division, and “.^” for exponentiation. For example, element-by-element multiplication of vectors of same dimension is accomplished by using the operator “.*”, as

$$\mathbf{c} = \mathbf{a} .* \mathbf{b} = \begin{bmatrix} a_1 b_1 \\ a_2 b_2 \\ a_3 b_3 \end{bmatrix}$$

Here \mathbf{a} , \mathbf{b} , and \mathbf{c} are column vectors with three elements. For addition and subtraction, element-by-element and usual matrix operations are the same. Other useful matrix operators are: $\mathbf{A}^2 = \mathbf{A} * \mathbf{A}$, $\mathbf{A}^{-1} = \text{inv}(\mathbf{A})$, determinant as $\text{det}(\mathbf{A})$, and transpose as \mathbf{A}' .

12.1.3 Optimization Toolbox

Optimization Toolbox must be installed in the computer in addition to MATLAB before it can be used. Table 12-1 shows the some “functions” available in the toolbox. Most of these optimization routines require M-files (stored in the current directory) containing a definition of the problem to be solved; several such files are presented and discussed later. Default optimization parameters are used extensively; however, they can be modified through an “options” command available in the program. The syntax of invoking an optimization function is generally of the form:

$$[\mathbf{x}, \text{FunValue}, \text{ExitFlag}, \text{Output}] = \text{fminX}(\text{'ObjFun'}, \dots, \text{options})$$

The left side of the statement represents the quantities returned by the “function.” These output arguments are described in Table 12-2. On the right side, fminX represents one of the functions given in Table 12-1. There can be several arguments for the function fminX ; e.g., starting values for the variables; upper and lower bounds for the variables; M-file names containing problem functions and their gradients; optimization algorithm related data; and so on. Use of this function is demonstrated in subsequent sections for various types of problems and conditions. For further explanation of various functions and commands, extensive on-line help is also available.

TABLE 12-1 Optimization Toolbox Functions

Type of problem	Formulation	Function
One-variable unconstrained minimization	Find $x \in [x_l, x_u]$ to minimize $f(x)$	<i>fminbnd</i>
Unconstrained minimization	Find \mathbf{x} to minimize $f(\mathbf{x})$	<i>fminunc</i> <i>fminsearch</i>
Constrained minimization	Find \mathbf{x} to minimize $f(\mathbf{x})$ subject to $\mathbf{N}\mathbf{x} = \mathbf{e}$, $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ $h_j = 0$, $j = 1$ to p $g_i(\mathbf{x}) \leq 0$, $i = 1$ to m $x_{il} \leq x_i \leq x_{iu}$	<i>fmincon</i>
Linear programming	Find \mathbf{x} to minimize $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$ subject to $\mathbf{N}\mathbf{x} = \mathbf{e}$, $\mathbf{A}\mathbf{x} \leq \mathbf{b}$	<i>linprog</i>
Quadratic programming	Find \mathbf{x} to minimize $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$ subject to $\mathbf{N}\mathbf{x} = \mathbf{e}$, $\mathbf{A}\mathbf{x} \leq \mathbf{b}$	<i>quadprog</i>

TABLE 12-2 Explanation of Output from Optimization Function

Argument	Description
\mathbf{x}	The solution vector or matrix found by the optimization function. If <code>ExitFlag</code> > 0 then \mathbf{x} is a solution, otherwise \mathbf{x} is the latest value from the optimization routine.
<code>FunValue</code>	Value of the objective function, <code>ObjFun</code> , at the solution \mathbf{x} .
<code>ExitFlag</code>	The exit condition for the optimization function. If <code>ExitFlag</code> is positive, then the optimization routine converged to a solution \mathbf{x} . If <code>ExitFlag</code> is zero, then the maximum number of function evaluations was reached. If <code>ExitFlag</code> is negative, then the optimization routine did not converge to a solution.
<code>Output</code>	The output vector contains several pieces of information about the optimization process. It provides the number of function evaluations (<code>Output.iterations</code>) and the name of the algorithm used to solve the problem (<code>Output.algorithm</code>), etc.

12.2 Unconstrained Optimum Design Problems

In this section, we first illustrate the use of *fminbnd* function for minimization of a function of single variable $f(x)$ with bounds on x as $x_l \leq x \leq x_u$. Then the use of function *fminunc* is illustrated for minimization of a function $f(\mathbf{x})$ of several variables. The M-files for the problems, containing extensive comments, are included to explain the use of these functions. Example 12.1 demonstrates use of the function *fminbnd* for functions of single variable.

EXAMPLE 12.1 Single-Variable Unconstrained Minimization

Find x to minimize $f(x) = 2 - 4x + e^x$, $-10 \leq x \leq 10$.

Solution. To solve the problem, we write an M-file that returns the objective function value. Then, we invoke the single-variable unconstrained minimization routine *fminbnd* through another M-file that is shown in Table 12-3. The file that evaluates the function, shown in Table 12-4, is called through *fminbnd*. Table 12-3 also shows results of the optimization process.

TABLE 12-3 M-File for Single Variable Unconstrained Minimizer *fminbnd* for Example 12.1

```
% All comments start with %
% File name: Example12_1.m
% Problem: minimize f(x) = 2 - 4x + exp(x)
clear all
% Set lower and upper bound for the design variable
Lb = -10; Ub = 10;
% Invoke single variable unconstrained optimizer fminbnd;
% The argument ObjFunction12_1 refers to the M-file that
% contains expression for the objective function
[x, FunVal, ExitFlag, Output] = fminbnd('ObjFunction12_1', Lb, Ub)
```

The output from the function is given as

x = 1.3863, FunVal = 0.4548, ExitFlag = 1 > 0 (i.e., minimum was found),
output = (iterations: 14, funcCount: 14, algorithm: golden section search, parabolic
interpolation.

TABLE 12-4 M-File for Objective Function for Example 12.1

```
% File name: ObjFunction12_1.m
% Example 12.1 Single variable unconstrained minimization
function f = ObjFunction12_1(x)
f = 2 - 4*x + exp(x);
```

Example 12.2 demonstrates the use of functions *fminsearch* and *fminunc* for multi-variable unconstrained optimization problems.

EXAMPLE 12.2 Multivariable Unconstrained Minimization

Consider a two variable problem of minimizing $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ starting from a point $\mathbf{x}^{(0)} = (-1.2, 1.0)$. Solve the problem using different algorithms available in the Optimization Toolbox.

Solution. The optimum solution for the problem is known as $\mathbf{x}^* = (1.0, 1.0)$ with $f(\mathbf{x}^*) = 0$ (Schittkowski, 1987). The syntax for functions *fminsearch* and *fminunc* used to solve a multivariable unconstrained optimization problem is given as follows:

```
[x, FunValue, ExitFlag, Output] = fminsearch ('ObjFun', x0, options)
[x, FunValue, ExitFlag, Output] = fminunc ('ObjFun', x0, options)
```

where ObjFun = name of the M-file that returns the function value and its gradient if programmed; x0 = starting values of design variables; and options = a command that

can be used to invoke various conditions for the optimization process. *fminsearch* uses the Simplex search method of Nelder-Mead, which does not require numerical or analytical gradient of the objective function.

Since *fminunc* does require this information and with the option `LargeScale` set to 'off', it uses the BFGS quasi-Newton method with a mixed quadratic and cubic line search procedures. The DFP formula, which approximates the inverse Hessian matrix, can be selected by setting the option `HessUpdate` to 'dfp'. The steepest descent method can be selected by setting option `HessUpdate` to 'steepdesc'. *fminsarch* is generally less efficient than *fminunc*. However, it can be effective for problems for which the gradient evaluation is expensive or not possible.

To solve this problem, we write an M-file that returns the objective function value. Then, the unconstrained minimization function *fminsearch* or *fminunc* is invoked through execution of another M-file shown in Table 12-5. The M-file for function and gradient evaluations is shown in Table 12-6. The gradient evaluation option can be omitted if its automatic evaluation by the finite difference method is desired. Three solution methods are used as shown in Table 12-5. All methods converge to the known solution.

TABLE 12-5 M-File for Unconstrained Optimization Routines for Example 12.2

```

% File name: Example12_2
% Rosenbruck valley function with analytical gradient of
% the objective function
clear all
x0 = [-1.2 1.0]'; % Set starting values
% Invoke unconstrained optimization routines

% 1. Nelder-Mead simplex method, fminsearch
% Set options: medium scale problem, maximum number of function evaluations
% Note that "." indicates that the text is continued on the next line
options = optimset('LargeScale', 'off', 'MaxFunEvals', 300);
[x1, FunValue1, ExitFlag1, Output1] =...
fminsearch ('ObjAndGrad12_2', x0, options)

% 2. BFGS method, fminunc, default option
% Set options: medium scale problem, maximum number of function evaluations,
% gradient of objective function
options = optimset('LargeScale', 'off', 'MaxFunEvals', 300,...
'GradObj', 'on');
[x2, FunValue2, ExitFlag2, Output2] =...
fminunc ('ObjAndGrad12_2', x0, options)

% 3. DFP method, fminunc, HessUpdate = dfp
% Set options: medium scale optimization, maximum number of function evaluation,
% gradient of objective function, DFP method
options = optimset('LargeScale', 'off', 'MaxFunEvals', 300,...
'GradObj', 'on', 'HessUpdate', 'dfp');
[x3, FunValue3, ExitFlag3, Output3] =...
fminunc ('ObjAndGrad12_2', x0, options)

```

TABLE 12-6 M-File for Objective Function and Gradient Evaluations for Example 12.2

```

% File name: ObjAndGrad12_2.m
% Rosenbrock valley function
function [f, df] = ObjAndGrad12_2(x)
% Re-name design variable x
x1 = x(1); x2 = x(2); %
% Evaluate objective function
f = 100*(x2 - x1^2)^2 + (1 - x1)^2;
% Evaluate gradient of the objective function
df(1) = -400*(x2-x1^2)*x1 - 2*(1-x1);
df(2) = 200*(x2-x1^2);

```

12.3 Constrained Optimum Design Problems

The general constrained optimization problem treated by the function *fmincon* is defined in Table 12-1. The procedure for invoking this function is the same as for the unconstrained problems except that an M-file containing the constraint functions must also be provided. If analytical gradient expressions are programmed in the objective function and constraint functions M-files, then these are declared through the “options” command. Otherwise, *fmincon* uses numerical gradient calculations based on the finite difference method. Example 12.3 shows the use of this function for an inequality constrained problem. Equalities if present can be included similarly.

EXAMPLE 12.3 Constrained Minimization Problem Using *Fmincon* in Optimization Toolbox

Solve the problem to

$$\text{minimize } f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \quad (\text{a})$$

subject to the constraints

$$g_1(\mathbf{x}) = 100 - (x_1 - 5)^2 - (x_2 - 5)^2 \leq 0 \quad (\text{b})$$

$$g_2(\mathbf{x}) = -82.81 - (x_1 - 6)^2 - (x_2 - 5)^2 \leq 0 \quad (\text{c})$$

$$13 \leq x_1 \leq 100, \quad 0 \leq x_2 \leq 100 \quad (\text{d})$$

Solution. The optimum solution for the problem is known as $\mathbf{x} = (14.095, 0.84296)$ and $f(\mathbf{x}^*) = -6961.8$ (Schittkowski, 1981). Three M-files for the problem are given in Tables 12-7 to 12-9. The file in Table 12-7 invokes the function *fmincon* with appropriate arguments and options. The file in Table 12-8 contains the cost function and its gradient expressions, and the file in Table 12-9 contains the constraint functions and their gradients. The problem is solved successfully and the final results are shown in Table 12-7.

TABLE 12-7 M-File for Constrained Minimizer *fmincon* for Example 12.3

```
% File name: Example12_3
% Constrained minimization with gradient expressions available
% Calls ObjAndGrad12_3 and ConstAndGrad12_3
clear all
% Set options; medium scale, maximum number of function evaluation,
% gradient of objective function, gradient of constraints, tolerances
% Note that three periods “...” indicate continuation on next line
options = optimset ('LargeScale', 'off', 'GradObj', 'on',...
    'GradConstr', 'on', 'TolCon', 1e-8, 'TolX', 1e-8);
% Set bounds for variables
Lb = [13; 0]; Ub = [100; 100];
% Set initial design
x0 = [20.1; 5.84];
% Invoke fmincon; four [ ] indicate no linear constraints in the problem
[x, FunVal, ExitFlag, Output] = ...
    fmincon('ObjAndGrad12_3', x0, [], [], [], [], Lb, ...
    Ub, 'ConstAndGrad12_3', options)
```

The output from the function is given as

Active Constraints: 5, 6 (i.e, g(1) and g(2))
x = (14.095, 0.843), FunVal = -6.9618e+003, ExitFlag = 1 > 0 (i.e., minimum was found), output = (iterations: 6, funcCount: 13, stepsize: 1, algorithm: medium scale: SQP, quasi-Newton, line-search).

Note that the “active constraints” command at the optimum solution are identified with their index counted as Lb, Ub, inequality constraints, and equality constraints. If ‘Display off’ is included in the “options” command, then the set of active constraints is not printed.

TABLE 12-8 M-File for Objective Function and Gradient Evaluations for Example 12.3

```
% File name: ObjAndGrad12_3.m
function [f, gf] = ObjAndGrad12_3(x)
% f returns value of objective function; gf returns objective function gradient
% Re-name design variables x
x1 = x(1); x2 = x(2);
% Evaluate objective function
f = (x1-10)^3 + (x2-20)^3;
% Compute gradient of objective function
if nargin > 1
gf(1,1) = 3*(x1-10)^2;
gf(2,1) = 3*(x2-20)^2;
end
```

TABLE 12-9 Constraint Functions and Their Gradients Evaluation M-File for Example 12.3

```
% File name: ConstAndGrad12_3.m
function [g, h, gg, gh] = ConstAndGrad12_3(x)
% g returns inequality constraints; h returns equality constraints
% gg returns gradients of inequalities; each column contains a gradient
% gh returns gradients of equalities; each column contains a gradient
% Re-name design variables
x1 = x(1); x2 = x(2);
% Inequality constraints
g(1) = 100-(x1-5)^2-(x2-5)^2;
g(2) = -82.81+(x1-6)^2 + (x2-5)^2;
% Equality constraints (none)
h = [ ];
% Gradients of constraints
if nargin > 2
gg(1,1) = -2*(x1-5);
gg(2,1) = -2*(x2-5);
gg(1,2) = 2*(x1-6);
gg(2,2) = 2*(x2-5);
gh = [ ];
end
```

12.4 Optimum Design Examples with MATLAB

12.4.1 Location of Maximum Shear Stress for Two Spherical Bodies in Contact

Project/Problem Statement There are many practical applications where two spherical bodies come into contact with each other as shown in Fig. 12-1. It is desired to determine the maximum shear stress and its location along the z axis for a given value of the Poisson's ratio of the material, $\nu = 0.3$.

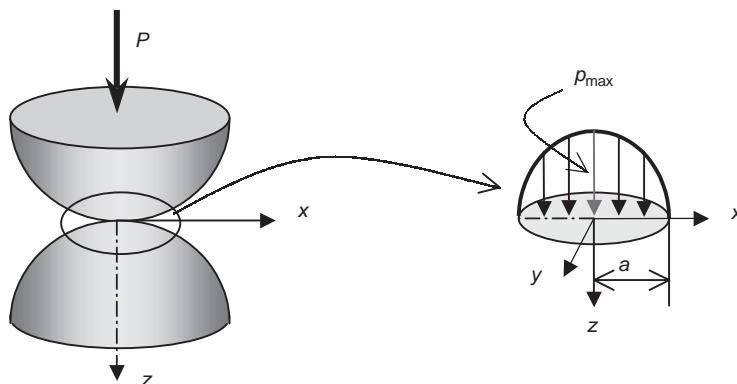


FIGURE 12-1 Spherical bodies in contact and pressure distribution on contact-patch.

Data and Information Collection The shear stress along the z -axis is calculated using the principal stresses as (Norton, 2000)

$$\sigma_{xz} = \frac{p_{\max}}{2} \left(\frac{1-2\nu}{2} + \frac{(1+\nu)\alpha}{\sqrt{1+\alpha^2}} - \frac{3}{2} \frac{\alpha^3}{\sqrt[3]{1+\alpha^2}} \right) \quad (a)$$

where $\alpha = z/a$ and a is the contact-patch radius as shown in Fig. 12-1. The maximum pressure occurs at the center of the contact patch a and is given as

$$p_{\max} = \frac{3P}{2\pi a^2} \quad (b)$$

It is well known that the peak shear stress does not occur at the contact surface but rather at a small distance below the surface. The subsurface location of the maximum shear stress is believed to be a significant factor in the surface fatigue failure called *pitting*.

Identification/Definition of Design Variables α is the only design variable for the problem.

Identification of a Criterion To Be Optimized The objective is to locate a point along the z axis where the shear stress is maximum. Transforming to the standard minimization form and normalizing with respect to p_{\max} , the problem becomes: find α to minimize

$$f(\alpha) = -\frac{\sigma_{xz}}{p_{\max}} \quad (c)$$

Identification of Constraints There are no constraints for the problem except bounds on the variable α taken as $0 \leq \alpha \leq 5$.

Solution The exact solution for the problem is given as

$$\left. \frac{\sigma_{xz}}{p_{\max}} \right|_{\max} = \frac{1}{2} \left(\frac{1-2\nu}{2} + \frac{2}{9} (1+\nu) \sqrt{2(1+\nu)} \right) \quad \text{at} \quad \alpha = \sqrt{\frac{2+2\nu}{7-2\nu}} \quad (d)$$

This is a single variable optimization problem with only lower and upper bounds on the variable. Therefore the function *fminbnd* in Optimization Toolbox can be used to solve the problem. Table 12-10 shows the M-file that invokes the function *fminbnd*, and Table 12-11 shows the M-file that evaluates the function to be minimized. The M-file also contains commands to plot the shear stress as a function of z , and it is shown in Fig. 12-2. The optimum solution is obtained as $\alpha = 0.6374$, $\text{FunVal} = -0.3329$ [$\alpha^* = 0.6374$, $f(\alpha^*) = -0.3329$] which matches the exact solution.

12.4.2 Column Design for Minimum Mass

Project/Problem Statement As noted in Section 2.7, columns are used as structural members in many practical applications. Many times such members are subjected to eccentric loads such as a jib crane. The problem is to design a minimum mass tubular column that is subjected to an eccentric load, as shown in Fig. 12-3. The cross section of the column is a hollow circular tube with R and t as the mean radius and wall thickness, respectively.

TABLE 12-10 M-File to Invoke Function *fminbnd* for Spherical Contact Problem

```
% File name: sphcont_opt.m
% Design variable: ratio of the max shear stress location to
% size of the contact patch
% Find location of the maximum shear stress along the z-axis
clear all
% Set lower and upper bound for the design variable
Lb = 0; Ub = 5;
% Plot normalized shear stress distribution along the z-axis in spherical contact
z = [Lb: 0.1: Ub]';
n = size(z);
for i = 1: n
    outz(i) = -sphcont_objf(z(i));
end
plot(z, outz); grid
xlabel('normalized depth z/a');
ylabel('normalized shear stress');
% Invoke the single-variable unconstrained optimizer
[alpha, FunVal, ExitFlag, Output] = fminbnd('sphcont_objf', Lb, Ub)
```

TABLE 12-11 M-File for Evaluation of Objective Function for the Spherical Contact Problem

```
% File name = sphcont_objf.m
% Location of max shear stress along z-axis for spherical contact problem
function f = sphcont_objf(alpha)
% f = - shear stress/max pressure
nu = 0.3; % Poisson's ratio
f = -0.5*( (1-2*nu)/2 + (1+nu)*alpha/sqrt(1+alpha^2) - ...
1.5*( alpha/sqrt(1+alpha^2) )^3 );
```

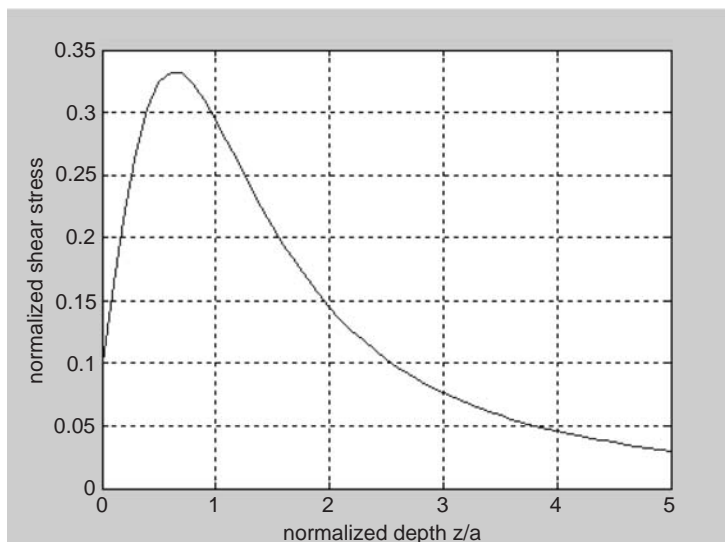


FIGURE 12-2 Normalized shear stress along the z-axis.

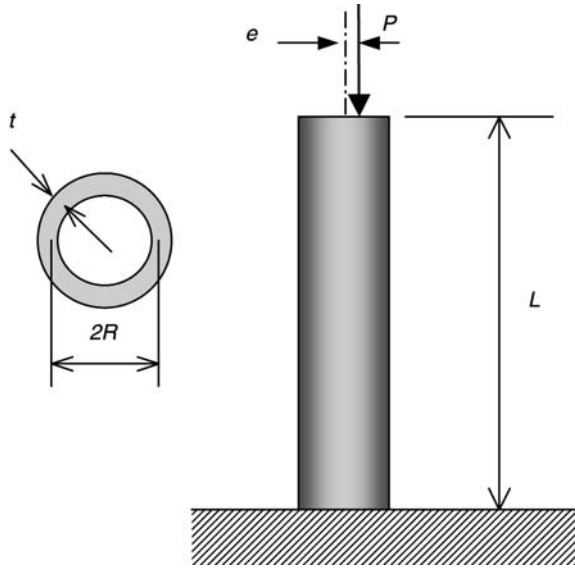


FIGURE 12-3 Configuration of vertical column with an eccentric load.

Data and Information Collection The data for the problem are given as

Load:	$P = 100 \text{ kN}$
Length:	$L = 5 \text{ m}$
Young's modulus:	$E = 210 \text{ GPa}$
Allowable stress:	$\sigma_a = 250 \text{ MPa}$
Eccentricity (2% of radius):	$e = 0.02R, \text{ m}$
Allowable deflection:	$\Delta = 0.25 \text{ m}$
Mass density:	$\rho = 7850 \text{ kg/m}^3$
Cross-sectional area:	$A = 2\pi R t, \text{ m}^2$
Moment of inertia:	$I = \pi R^3 t, \text{ m}^4$
Distance to the extreme fiber:	$c = R + \frac{1}{2}t, \text{ m}$

An analysis of the structure yields the following design equations:

$$\text{Normal stress: } \sigma = \frac{P}{A} \left[1 + \frac{ec}{k^2} \sec \left(\frac{L}{k} \sqrt{\frac{P}{EA}} \right) \right], \quad k^2 = \frac{I}{A} \quad (\text{a})$$

$$\text{Buckling load: } P_{cr} = \frac{\pi^2 EI}{4L^2} \quad (\text{b})$$

$$\text{Deflection: } \delta = e \left[\sec \left(L \sqrt{\frac{P}{EI}} \right) - 1 \right] \quad (\text{c})$$

Identification/Definition of Design Variables Two design variables for the problem are defined as

R = mean radius of the tube, m
 t = wall thickness, m

Identification of a Criterion To Be Optimized The objective is to minimize the mass of the column which is given as

$$f(\mathbf{x}) = \rho LA = (7850)(5)(2\pi Rt), \text{ kg} \quad (\text{d})$$

Identification of Constraints Constraints for the problem are on performance of the structure, maximum radius to thickness ratio, and bounds on the radius and thickness:

$$\text{Stress constraint:} \quad \sigma \leq \sigma_a \quad (\text{e})$$

$$\text{Buckling load constraint:} \quad P \leq P_{cr} \quad (\text{f})$$

$$\text{Deflection constraint:} \quad \delta \leq \Delta \quad (\text{g})$$

$$\text{Radius/thickness constraint:} \quad \frac{R}{t} \leq 50 \quad (\text{h})$$

$$\text{Bounds on the variables:} \quad 0.01 \leq R \leq 1, \quad 0.005 \leq t \leq 0.2 \quad (\text{i})$$

Solution Let us redefine the design variables and other parameters for MATLAB as

$$x_1 = R, \quad x_2 = t \quad (\text{j})$$

$$c = x_1 + \frac{1}{2}x_2, \quad e = 0.02x_1 \quad (\text{k})$$

$$A = 2\pi x_1 x_2, \quad I = \pi x_1^3 x_2, \quad k^2 = \frac{I}{A} = \frac{x_1^2}{2} \quad (\text{l})$$

All constraints are normalized and rewritten using redefined design variables. Therefore the optimization problem is stated in the standard form as follows:

$$\text{minimize } f(\mathbf{x}) = 2\pi(5)(7850)x_1 x_2 \quad (\text{m})$$

subject to

$$g_1(\mathbf{x}) = \frac{P}{2\pi x_1 x_2 \sigma_a} \left[1 + \frac{2 \times 0.02(x_1 + 0.5x_2)}{x_1} \sec \left(\frac{\sqrt{2}L}{x_1} \sqrt{\frac{P}{E(2\pi x_1 x_2)}} \right) \right] - 1 \leq 0 \quad (\text{n})$$

$$g_2(\mathbf{x}) = 1 - \frac{\pi^2 E(2\pi x_1^3 x_2)}{4L^2 P} \leq 0 \quad (\text{o})$$

$$g_3(\mathbf{x}) = \frac{0.02x_1}{\Delta} \left[\sec \left(L \sqrt{\frac{P}{E(\pi x_1^3 x_2)}} \right) - 1 \right] - 1 \leq 0 \quad (\text{p})$$

$$g_4(\mathbf{x}) = \frac{x_1}{50x_2} - 1 \leq 0 \quad (\text{q})$$

$$0.01 \leq x_1 \leq 1, \quad 0.005 \leq x_2 \leq 0.2 \quad (\text{r})$$

The problem is solved using the *fmincon* function in the Optimization Toolbox. Table 12-12 shows the M-file for invoking this function and setting various options for the optimization process. Tables 12-13 and 12-14 show the M-files for the objective and constraint functions, respectively. Note that analytical gradients are not provided for the problem functions.

The output from the function is given as

Active Constraints: 2, 5, i.e., the lower limit for thickness and $g(1)$.

$x = (0.0537, 0.0050)$, FunVal = 66.1922, ExitFlag = 1, Output = (iterations: 31, funcCount: 149, stepsize: 1, algorithm: medium-scale: SQP, Quasi-Newton, line-search).

TABLE 12-12 M-File for Invoking Minimization Function for Column Design Problem

```

% File name = column_opt.m
clear all
% Set options
options = optimset ('LargeScale', 'off', 'TolCon', 1e-8, 'TolX', 1e-8);
% Set the lower and upper bounds for design variables
Lb = [0.01 0.005]; Ub = [1 0.2];
% Set initial design
x0 = [1 0.2];
% Invoke the constrained optimization routine, fmincon
[x, FunVal, ExitFlag, Output] = ...
    fmincon('column_objf', x0, [], [], [], [], Lb, Ub, 'column_conf', options)

```

TABLE 12-13 M-File for Objective Function for Minimum Mass Column Design Problem

```

% File name = column_objf.m
% Column design
function f = column_objf (x)
% Rename design variables
x1 = x(1); x2 = x(2);
% Set input parameters
L = 5.0; % length of column (m)
rho = 7850; % density (kg/m^3)
f = 2*pi*L*rho*x1*x2; % mass of the column

```

TABLE 12-14 M-File for Constraint Functions for the Column Design Problem

```

% File name = column_conf.m
% Column design
function [g, h] = column_conf (x)
x1 = x(1); x2 = x(2);
% Set input parameters
P = 50000; % loading (N)
E = 210e9; % Young's modulus (Pa)
L = 5.0; % length of the column (m)
Sy = 250e6; % allowable stress (Pa)
Delta = 0.25; % allowable deflection (m)
% Inequality constraints
g(1) = P/(2*pi*x1*x2)*(1 + ...
    2*0.02*(x1+x2/2)/x1*sec( 5*sqrt(2)/x1*sqrt(P/E/(2*pi*x1*x2)) ) )/Sy - 1;
g(2) = 1 - pi^3*E*x1^3*x2/4/L^2/P;
g(3) = 0.02*x1*( sec( L*sqrt( P/(pi*E*x1^3*x2) ) ) - 1 )/Delta - 1;
g(4) = x1/x2/50 - 1;
% Equality constraint (none)
h = [];

```

12.4.3 Flywheel Design for Minimum Mass

Project/Problem Statement Shafts are used in practical applications to transfer torque from a source point to another point to achieve desired operations. However, the torque to be transferred can fluctuate causing variations in the angular speed of the shaft which is not

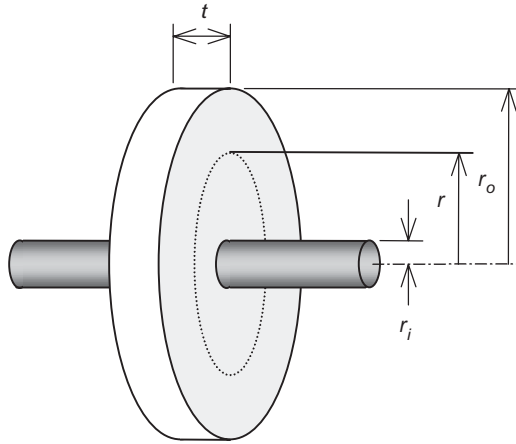


FIGURE 12-4 Flywheel-shaft system.

desirable. Flywheels are used on the shaft to smooth out these variations of the speed (Norton, 2000; Shigley and Mischke, 2001). The purpose of this project is to design a flywheel to smooth out variations in the speed of a solid shaft of radius r_i . The flywheel-shaft system is shown in Fig. 12-4. The input torque function, which varies during a cycle, is shown in Fig. 12-5. This torque variation about its average value, as a function of the shaft angle for one 360° cycle, is shown there. The kinetic energy due to this variation is obtained by integrating the torque pulse above and below its average value during the cycle and is given as $E_k = 26,105$ in·lb. The shaft is rotating at a nominal angular speed of $\omega = 800$ rad/s.

Data and Information Collection The one cycle of torque variation shown in Fig. 12-5 is assumed to be repetitive and thus representative of the steady-state condition. The desired coefficient of fluctuation is assumed to be 0.05 (C_f). The coefficient of fluctuation represents the ratio of variation of angular velocity to the nominal angular velocity: $C_f = (\omega_{\max} - \omega_{\min})/\omega$. The system is assumed to be in continuous operation with minimal start-stop cycles. The minimum mass moment of inertia for the flywheel is determined using the required change in kinetic energy, E_k , specified earlier, as

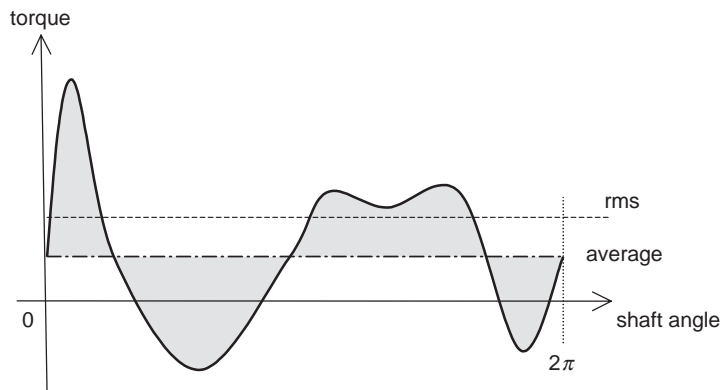


FIGURE 12-5 Fluctuation about average value of input torque for one cycle.

$$I_s = \frac{E_k}{C_f \omega^2} = \frac{26,105}{0.05(800)^2} = 0.816 \text{ lb} \cdot \text{in} \cdot \text{s}^2 \quad (\text{a})$$

The design data and equations needed to formulate the minimum mass flywheel problem are given as

Specific weight:	$\gamma = 0.28 \text{ lb/in}^3$	
Gravitational constant:	$g = 386 \text{ in/s}^2$	
Yield stress:	$S_y = 62,000 \text{ psi}$	
Nominal angular velocity:	$\omega = 800 \text{ rad/s}$	
Poisson's ratio:	$\nu = 0.28$	
Inner radius of flywheel:	$r_i = 1.0 \text{ in}$	
Outer radius of flywheel:	$r_o, \text{ in}$	
Thickness of flywheel:	$t, \text{ in}$	
Mass moment of inertia of flywheel:	$I_m = \frac{\pi \gamma}{2 g} (r_o^4 - r_i^4) t, \text{ lb} \cdot \text{in} \cdot \text{s}^2$	(b)
Tangential stress in flywheel at radius r :	$\sigma_t = \frac{\gamma}{g} \omega^2 \frac{3+\nu}{8} \left(r_i^2 + r_o^2 + \frac{r^2 r_o^2}{r^2} - \frac{1+3\nu}{3+\nu} r^2 \right), \text{ psi}$	(c)
Radial stress in flywheel at radius r :	$\sigma_r = \frac{\gamma}{g} \omega^2 \frac{3+\nu}{8} \left(r_i^2 + r_o^2 - \frac{r^2 r_o^2}{r^2} - r^2 \right), \text{ psi}$	(d)
von Mises stress:	$\sigma' = \sqrt{\sigma_r^2 - \sigma_r \sigma_t + \sigma_t^2}, \text{ psi}$	(e)

Identification/Definition of Design Variables The two design variables for the problem are defined as

$$r_o = \text{outer radius of the flywheel, in}$$

$$t = \text{thickness of the flywheel, in}$$

Identification of a Criterion To Be Optimized The objective of the project is to design a flywheel of minimum mass. Since mass is proportional to the material volume, it is desired to minimize the volume of the flywheel, which is given as

$$f = \pi(r_o^2 - r_i^2)t, \text{ in}^3 \quad (\text{f})$$

Identification of Constraints Performance and other constraints are expressed as

Mass moment of inertia requirement:	$I_m \geq I_s$	(g)
von Mises stress constraint:	$\sigma' \leq \frac{1}{2} S_y$	(h)
Limits on design variables:	$4.5 \leq r_o \leq 9.0, \quad 0.25 \leq t \leq 1.25$	(i)

Solution The problem is solved using the *fmincon* function in the Optimization Toolbox. Table 12-15 shows the M-file that invokes the *fmincon* function for the flywheel problem. Table 12-16 shows the M-file that calculates the objective function for the problem. Table 12-17 shows the M-file for calculation of constraints for the problem. Note that the von Mises stress constraint is imposed at the point of maximum stress. Therefore this maximum is calculated using the *fmincon* function itself. Table 12-18 shows the M-file that calculates the von Mises stress. This file is called by the constraint evaluation file. Also note that all constraints are entered in the normalized “ \leq ” form. The solution and other output from the function are given as

$$\text{Active constraints are 2 and 5, i.e., lower bound on thickness and } g(1)$$

$$r_o^* = 7.3165 \text{ in}, t^* = 0.25 \text{ in}, f^* = 13.1328, \text{ Output} = [\text{iterations: } 8, \text{ funcCount: } 37]$$

TABLE 12-15 M-File to Invoke Constrained Minimization Routine for Flywheel Design Problem

```
% File name = flywheel_opt.m
% Flywheel design
% Design variables: outside radius (ro), and thickness (t)
clear all
% Set options
options = optimset('LargeScale', 'off');
% Set limits for design variables
Lb = [4.5, 0.25]; % lower limit
Ub = [9, 1.25]; % upper limit
% Set initial design
x0 = [6, 1.0];
% Set radius of shaft
ri = 1.0;
[x, FunVal, ExitFlag, Output] = ...
fmincon('flywheel_objf', x0, [], [], [], [], Lb, Ub, 'flywheel_conf', options, ri)
```

TABLE 12-16 M-File for Objective Function for Flywheel Design Problem

```
% File name = flywheel_objf.m
% Objective function for flywheel design problem
function f = flywheel_objf(x, ri)
% Rename the design variables x
ro = x(1);
t = x(2);
f = pi*(ro^2 - ri^2)*t; % volume of flywheel
```

TABLE 12-17 M-File for Constraint Functions for Flywheel Design Problem

```
% Constraint functions for flywheel design problem
function [g, h] = flywheel_conf(x, ri)
% Rename design variables x
ro = x(1);
t = x(2);
% Constraint limits
Is = 0.816; % mass moment of inertia
Sy = 62000; % yield strength
% Normalized inequality constraints
g(1) = 1 - pi/2*(0.28/386)*(ro^4 - ri^4)*t/Is;
% Evaluate maximum von Mises stress
options = [];
[alpha, vonMS] = fminbnd('flywheel_vonMs', ri, ro, options, ri, ro);
g(2) = -vonMS/(0.5*Sy) - 1;
% Equality constraint (none)
h = [];
```

TABLE 12-18 M-File for Computation of Maximum von Mises Stress at Current Design

```
% File name = flywheel_vonMS.m
% von Mises stress
function vonMS = flywheel_vonMS (x, ri, ro)
temp = (0.28/386)*(800)^2*(3+0.28)/8;
% Tangential stress
st = temp*(ri^2 + ro^2 + ri^2*ro^2/x^2 - (1+3*0.28)/(3+0.28)*x^2 );
% Radial stress
sr = temp*(ri^2 + ro^2 - ri^2*ro^2/x^2 - x^2); % radial stress
vonMS = -sqrt(st^2 - st*sr + sr^2); % von Mises stress
```

Exercises for Chapter 12*

Formulate and solve the following problems.

- 12.1 Exercise 3.34 12.2 Exercise 3.35 12.3 Exercise 3.36
 12.4 Exercise 3.50 12.5 Exercise 3.51 12.6 Exercise 3.52
 12.7 Exercise 3.53 12.8 Exercise 3.54

- 12.9 Consider the cantilever beam-mass system shown in Fig. E12-9. Formulate and solve the minimum weight design problem for the rectangular cross section so that the fundamental vibration frequency is larger than 8rad/s and the cross-sectional dimensions satisfy the limitations

$$0.5 \leq b \leq 1.0, \quad \text{in}$$

$$0.2 \leq h \leq 2.0, \quad \text{in}$$

Use a nonlinear programming algorithm to solve the problem. Verify the solution graphically and trace the history of the iterative process on the graph of the problem. Let the starting point be (0.5, 0.2). The data and various equations for the problem are as follows:

Fundamental vibration frequency	$\omega = \sqrt{k_e/m}$ rad/s
Equivalent spring constant, k_e	$\frac{1}{k_e} = \frac{1}{k} + \frac{L^3}{3EI}$
Mass attached to the spring	$m = W/g$
Weight attached to the spring	$W = 50\text{lb}$
Length of the beam	$L = 12\text{in}$
Modulus of elasticity	$E = (3.00\text{E}+07)\text{psi}$
Spring constant	$k = 10\text{lb/in}$
Moment of inertia	I, in^4
Gravitational constant	$g, \text{in/s}^2$

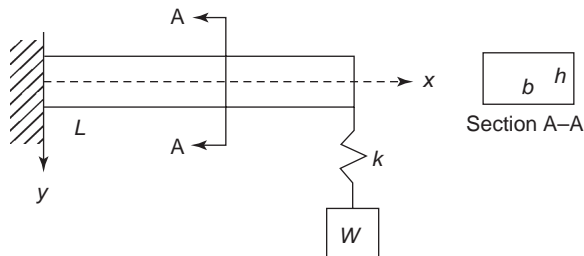


FIGURE E12-9 Cantilever beam with spring-mass at the free end.

12.10 A prismatic steel beam with symmetric I cross section is shown in Fig. E12-10. Formulate and solve the minimum weight design problem subject to the following constraints:

1. The maximum axial stress due to combined bending and axial load effects should not exceed 100 MPa.
2. The maximum shear stress should not exceed 60 MPa.
3. The maximum deflection should not exceed 15 mm.
4. The beam should be guarded against lateral buckling.
5. Design variables should satisfy the limitations $b \geq 100$ mm, $t_1 \leq 10$ mm, $t_2 \leq 15$ mm, $h \leq 150$ mm.

Solve the problem using a numerical optimization method, and verify the solution using KKT necessary conditions for the following data:

Modulus of elasticity $E = 200$ GPa
 Shear modulus $G = 70$ GPa
 Load $P = 70$ kN
 Load angle $\theta = 45^\circ$
 Beam length $L = 1.5$ m

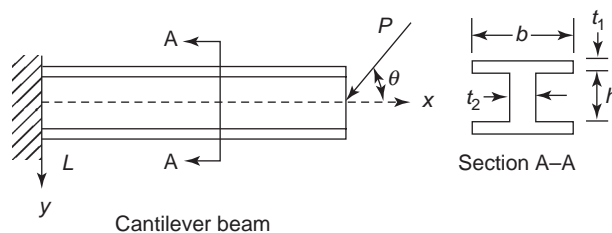


FIGURE E12-10 Cantilever I beam. Design variables: b , t_1 , t_2 , and h .

12.11 *Shape optimization of a structure.* The design objective is to determine the shape of the three-bar structure shown in Fig. E12-11 to minimize its weight (Corcoran, 1970). The design variables for the problem are the member cross-sectional areas A_1 , A_2 , and A_3 and the coordinates of nodes A, B, and C (note that x_1 , x_2 , and x_3 have positive values in the figure; the final values can be positive or negative), so that the truss is as light as possible while satisfying the stress constraints due to the following three loading conditions:

Cond. No. j	Load P_j (lb)	Angle θ_j (degrees)
1	40,000	45
2	30,000	90
3	20,000	135

The stress constraints are written as

$$\begin{aligned} -5000 &\leq \sigma_{1j} \leq 5000, \text{ psi} \\ -20,000 &\leq \sigma_{2j} \leq 20,000, \text{ psi} \\ -5000 &\leq \sigma_{3j} \leq 5000, \text{ psi} \end{aligned}$$

where $j = 1, 2, 3$ represents the index for the three loading conditions and the stresses are calculated from the following expressions:

$$\sigma_{1j} = \frac{E}{L_1} [u_j \cos \alpha_1 + v_j \sin \alpha_1] = \frac{E}{L_1^2} (u_j x_1 + v_j L)$$

$$\sigma_{2j} = \frac{E}{L_2} [u_j \cos \alpha_2 + v_j \sin \alpha_2] = \frac{E}{L_2^2} (u_j x_2 + v_j L)$$

$$\sigma_{3j} = \frac{E}{L_3} [u_j \cos \alpha_3 + v_j \sin \alpha_3] = \frac{E}{L_3^2} (-u_j x_3 + v_j L)$$

where $L = 10$ in and

$$L_1 = \text{length of member 1} = \sqrt{L^2 + x_1^2}$$

$$L_2 = \text{length of member 2} = \sqrt{L^2 + x_2^2}$$

$$L_3 = \text{length of member 3} = \sqrt{L^2 + x_3^2}$$

and u_j and v_j are the horizontal and vertical displacements for the j th loading condition determined from the following linear equations:

$$\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{bmatrix} u_j \\ v_j \end{bmatrix} = \begin{bmatrix} P_j \cos \theta_j \\ P_j \sin \theta_j \end{bmatrix}, \quad j = 1, 2, 3$$

where the stiffness coefficients are given as ($E = 3.0E+07$ psi)

$$k_{11} = E \left(\frac{A_1 x_1^2}{L_1^3} + \frac{A_2 x_2^2}{L_2^3} + \frac{A_3 x_3^2}{L_3^3} \right)$$

$$k_{12} = E \left(\frac{A_1 L x_1}{L_1^3} + \frac{A_2 L x_2}{L_2^3} - \frac{A_3 L x_3}{L_3^3} \right) = k_{21}$$

$$k_{22} = E \left(\frac{A_1 L^2}{L_1^3} + \frac{A_2 L^2}{L_2^3} + \frac{A_3 L^2}{L_3^3} \right)$$

Formulate the design problem and find the optimum solution starting from the point

$$A_1 = 6.0, \quad A_2 = 6.0, \quad A_3 = 6.0$$

$$x_1 = 5.0, \quad x_2 = 0.0, \quad x_3 = 5.0$$

Compare the solution with that given later in Table 14.7.

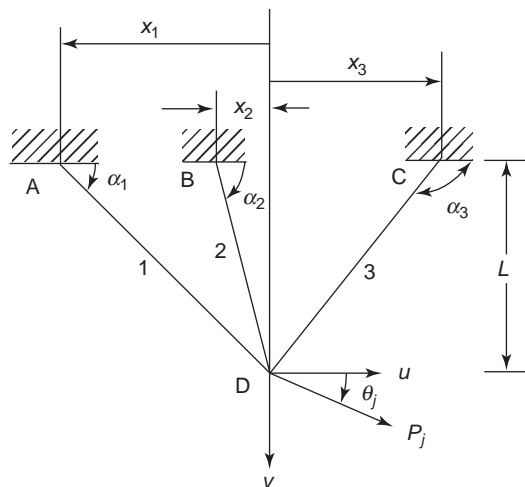


FIGURE E12-11 Three-bar structure–shape optimization.

12.12 *Design synthesis of a nine-speed gear drive.* The arrangement of a nine-speed gear train is shown in Fig. E12-12. The objective of the synthesis is to find the size of all gears from the mesh and speed ratio equations such that the size of the largest gears are kept to a minimum (Osman *et al.*, 1978). Because of the mesh and speed ratio equations, it is found that only the following three independent parameters need to be selected:

$$x_1 = \text{gear ratio, } d/a$$

$$x_2 = \text{gear ratio, } e/a$$

$$x_3 = \text{gear ratio, } j/a$$

Because of practical considerations, it is found that the minimization of $|x_2 - x_3|$ results in the reduction of the cost of manufacturing the gear drive.

The gear sizes must satisfy the following mesh equations:

$$\phi^2 x_1 (x_1 + x_3 - x_2) - x_2 x_3 = 0$$

$$\phi^3 x_1 - x_2 (1 + x_2 - x_1) = 0$$

where ϕ is the step ratio in speed. Find the optimum solution for the problem for two different values of ϕ as $\sqrt{2}$ and $(2)^{1/3}$.

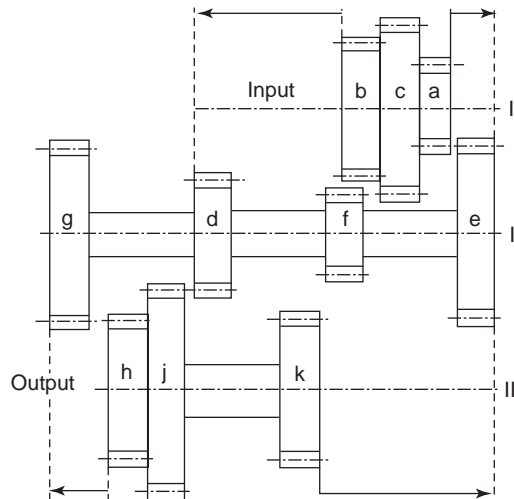


FIGURE E12-12 Schematic arrangement of a nine-speed gear train.

13 Interactive Design Optimization

Upon completion of this chapter, you will be able to:

- Explain the kind of interaction possible with an optimization algorithm
- Use optimization software in an interactive mode, if it offers such facilities
- Use an optimization algorithm in a manual interactive mode, if appropriate

Optimization techniques can—to some extent—automate the tedious trial-and-error aspects of the design process, thus allowing the engineer to concentrate on the more creative aspects of designing systems. They harness a computer's speed with computational algorithms to methodically generate the efficient (not just satisfactory) designs that are needed in today's competitive world. Optimization techniques can be applied to virtually any engineering design situation, such as the design of aircraft structures, buildings, automotive structures, engine components, heat exchangers, land developments, water reclamation projects, chemical processes, electronic circuits, and many more.

Most numerical methods for design optimization involve considerable repetitive calculations. They must be transcribed into proper computer software. Performance and robustness of the methods are affected by the round-off and truncation errors in computer calculations. Most optimization algorithms are proved to converge only in the limit, i.e., there is no algorithm for general optimization problems that is proved to converge in a finite number of iterations. In addition, algorithms lack preciseness in their computational steps because a given algorithm can be implemented in several different ways depending on the programmer's knowledge, experience, and preferences. In addition, the performance of software can differ when it is installed on different computers, or when different compilers are used on the same computer. All these difficulties and the lack of preciseness point to the need to somehow monitor the progress of an algorithm toward the solution and interact with it if needed.

In order to monitor progress of the optimum design process, proper hardware and software are needed. *The software must have proper interactive facilities so the designer can change the course of the design process if necessary.* The design information must be displayed in a comprehensible form. Proper help facilities should also be available. The graphical display of various data and information can facilitate the interactive decision-making process, so it should be available (Arora and Tseng, 1988).

In this chapter, we describe *the interactive design optimization process*. The role of designer interaction and algorithms for interaction are described, especially for advanced users who would prefer to interact with the optimization process. Desired interactive capabilities and decision-making facilities are discussed and simple examples are used to demonstrate their use in the design process. These discussions essentially lay out the specifications for an interactive design optimization software.

13.1 Role of Interaction in Design Optimization

13.1.1 What Is Interactive Design Optimization?

In Chapter 1 we described the engineering design process. The differences between the conventional and the optimum design process were explained. The optimum design process requires sophisticated computational algorithms. However, most algorithms have some uncertainties in their computational steps. Therefore, it is sometimes prudent to interactively monitor their progress and guide the optimum design process. *Interactive design optimization algorithms are based on utilizing the designer's input during the iterative process. They are in some sense open-ended algorithms in which the designer can specify what needs to be done depending on the current design conditions. They must be implemented into interactive software that can be interrupted during the iterative process and that can report the status of the design to the user.* Relevant data and conditions must be displayed at the designer's command at a graphics workstation. Various options should be available to the designer to facilitate decision making and change design data. It should be possible to restart or terminate the process. With such facilities, designers have complete control over the design optimization process. They can guide it to obtain better designs and ultimately the best design.

It is clear that for interactive design optimization, proper algorithms must be implemented into highly flexible and user-friendly software. It must be possible for the designer to interact with the algorithm and change the course of its calculations. We describe later in Section 13.2 algorithms that are suitable for designer interaction. Figure 13-1 is a conceptual flow diagram for the interactive design optimization process. It is a modification of Fig. 1-2 in which an interactive block has been added. The designer interacts with the design process through this block. We shall discuss the desired interactive capabilities and their use later in this chapter.

13.1.2 Role of Computers in Interactive Design Optimization

As we have discussed earlier, *the conventional trial-and-error design philosophy is changing with the emergence of fast computers and computational algorithms*. The new design methodology is characterized by the phrase *model and analyze*. Once the design problem is properly formulated, numerical methods can be used to optimize the system. The methods are iterative and generate a sequence of design points before converging to the optimum solution. They are best suited for computer implementation to exploit the speed of computers for performing repetitive calculations.

It is extremely important to select only robust optimization algorithms for practical applications. Otherwise, failure of the design process will undoubtedly result in the waste of computer resources and, more importantly, the loss of the designer's time and morale.

An optimization algorithm involves a limiting process, because some parameters go to zero or infinity as the optimum design is approached. The representation of such limiting processes is difficult in *computer implementation* as it may lead to underflow or overflow. In other words, the limiting processes can never be satisfied exactly on a computer and quantities such as zero and infinity must be redefined as very small and large numbers, respectively, on the computer. These quantities are relative and machine-dependent.

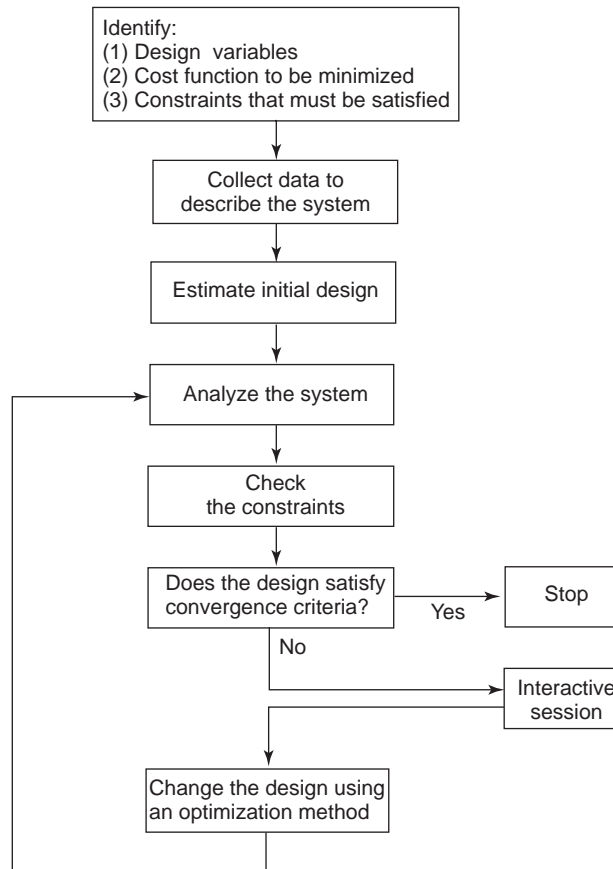


FIGURE 13-1 Interactive optimum design process.

Often, the proof of convergence or rate of convergence of an iterative optimization algorithm is based on *exact arithmetic* and under restrictive conditions. Thus, the theoretical behavior of an algorithm may no longer be valid in practice because of inexact arithmetic causing round-off and truncation errors in computer representation of numbers. This discussion highlights the fact that proper coding and interactive monitoring of theoretically convergent algorithms are equally important.

13.1.3 Why Interactive Design Optimization?

The design process can be quite complex. Often the problem cannot be stated in a precise form for complete analysis and there are uncertainties in the design data. The solution to the problem need not exist. On many occasions, *the formulation of the problem must be developed as part of the design process*. Therefore, it is neither desirable nor useful to optimize an inexact problem to the end in a batch environment. It would be a complete waste of valuable resources to find out at the end that wrong data were used or a constraint was inadvertently omitted. It is desirable to have an interactive algorithm and software capable of designer interaction. Such a capability can be extremely useful in a practical design environment because not only can better designs be obtained, but more *insights into the problem behavior can be gained*. *The problem formulation can be refined*, and inadequate and absurd designs can be avoided. We shall describe some interactive algorithms and other suitable capabilities to demonstrate the usefulness of designer interaction in the design process.

13.2 Interactive Design Optimization Algorithms

It is clear from the preceding discussion that for a useful interactive capability, proper algorithms must be implemented into well-designed software. Some optimization algorithms are not suitable for designer interaction. For example, the constrained steepest descent method of Section 10.5 and the quasi-Newton method of Section 11.4 are not suitable for the interactive environment. Their steps are in a sense closed-ended allowing little opportunity for the designer to change course from the iterative design process. However, it turns out that the *QP subproblem and the basic concepts discussed there can be utilized to devise algorithms suitable for the interactive environment*. We shall describe these algorithms and illustrate them with examples.

Depending on the design condition at the current iteration, the *designer may want to ask any of the following four questions*:

1. If the current design is feasible but not optimum, can the cost function be reduced by γ percent?
2. If the starting design is infeasible, can a feasible design be obtained at any cost?
3. If the current design is infeasible, can a feasible design be obtained without increasing the cost?
4. If the current design is infeasible, can a feasible design be obtained with only δ percent penalty on the cost?

We shall describe algorithms to answer these questions. It will be seen that the algorithms are conceptually quite simple and easy to implement. As a matter of fact, they are modifications of the constrained steepest descent (CSD) and quasi-Newton methods of Sections 10.5 and 11.4. It should also be clear that if interactive software with commands to execute the foregoing steps is available, the designer can actually use the commands to guide the process to successively better designs and ultimately an optimum design.

13.2.1 Cost Reduction Algorithm

A subproblem for the cost reduction algorithm can be defined with or without the approximate Hessian \mathbf{H} . Without Hessian updating, the problem is defined in Eqs. (10.25) and (10.26) and, with Hessian updating, it is defined in Eqs. (11.48) to (11.50). Although Hessian updating can be used, we shall define the cost reduction subproblem without it to keep the discussion and the presentation simple. Since the cost reduction problem is solved from a feasible or almost feasible point, the right side vector \mathbf{e} in Eq. (10.26) is zero. Thus, the cost reduction QP subproblem is defined as

$$\text{minimize } \bar{f} = \mathbf{c}^T \mathbf{d} + 0.5 \mathbf{d}^T \mathbf{d} \quad (13.1)$$

$$\text{subject to } \mathbf{N}^T \mathbf{d} = \mathbf{0} \quad (13.2)$$

$$\mathbf{A}^T \mathbf{d} \leq \mathbf{b} \quad (13.3)$$

The columns of matrices \mathbf{N} and \mathbf{A} contain gradients of equality and inequality constraints, respectively, and \mathbf{c} is the gradient of the cost function. Equation (13.2) gives the dot product of \mathbf{d} with all the columns of \mathbf{N} as zero. Therefore, \mathbf{d} is orthogonal to the gradients of all the equality constraints. Since gradients in the matrix \mathbf{N} are normal to the corresponding constraint surfaces, the search direction \mathbf{d} lies in a plane tangent to the equality constraints. The right side vector \mathbf{b} for the inequality constraints in Eq. (13.3) contains zero elements corresponding to the active constraints and positive elements corresponding to the inactive con-

straints. If an active constraint remains satisfied at the equality (i.e., $\mathbf{a}^{(i)} \cdot \mathbf{d} = 0$), the direction \mathbf{d} is in a plane tangent to that constraint. Otherwise, it must point into the feasible region for the constraint.

The QP subproblem defined in Eqs. (13.1) to (13.3) can incorporate the potential constraint strategy as explained in Section 11.1. The subproblem can be solved for the cost reduction direction by any of the available subroutines cited in Section 11.2. In the example problems, however, we shall solve the QP subproblem using KKT conditions. We shall call this procedure of reducing cost from a feasible point the *cost reduction (CR) algorithm*.

After the direction has been determined, the step size can be calculated by a line search on the proper descent function. Or, we can require a certain reduction in the cost function and determine the step size that way. For example, we can require a fractional reduction γ in the cost function (for a 5 percent reduction, $\gamma = 0.05$), and calculate a step size based on it. Let α be the step size along \mathbf{d} . Then the first-order change in the cost using a linear Taylor's expansion is given as $\alpha|\mathbf{c} \cdot \mathbf{d}|$. Equating this to the required reduction in cost $|\gamma f|$, the step size is calculated as

$$\alpha = \frac{|\gamma f|}{|\mathbf{c} \cdot \mathbf{d}|} \quad (13.4)$$

Note that $\mathbf{c} \cdot \mathbf{d}$ should not be zero in Eq. (13.4) to give a reasonable step size. The cost reduction step is illustrated in Example 13.1.

EXAMPLE 13.1 Cost Reduction Step

Consider the design optimization problem

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) = x_1^2 - 3x_1x_2 + 4.5x_2^2 - 10x_1 - 6x_2 \\ &\text{subject to} \quad \quad \quad x_1 - x_2 \leq 3 \\ &\quad \quad \quad \quad \quad \quad x_1 + 2x_2 \leq 12 \\ &\quad \quad \quad \quad \quad \quad x_1, x_2 \leq 0 \end{aligned}$$

From the feasible point (4, 4), calculate the cost reduction direction and the new design point requiring a cost reduction of 10 percent.

Solution. The constraints can be written in the standard form as

$$\begin{aligned} g_1 &= \frac{1}{3}(x_1 - x_2) - 1.0 \leq 0 \\ g_2 &= \frac{1}{12}(x_1 + 2x_2) - 1.0 \leq 0 \\ g_3 &= -x_1 \leq 0 \\ g_4 &= -x_2 \leq 0 \end{aligned}$$

The optimum solution for the problem is calculated using the KKT conditions as

$$\mathbf{x}^* = (6, 3); \quad \mathbf{u}^* = (17, 16, 0, 0); \quad f(\mathbf{x}^*) = -55.5$$

At the given point (4, 4),

$$f(4, 4) = -24$$

$$g_1 = -1.0 < 0 \text{ (inactive)}$$

$$g_2 = 0 \text{ (active)}$$

$$g_3 = -4.0 < 0 \text{ (inactive)}$$

$$g_4 = -4.0 < 0 \text{ (inactive)}$$

Therefore, constraint g_2 is active, and all the others are inactive. The cost function is much larger than the optimum value. The constraints for the problem are plotted in Fig. 13-2. The feasible region is identified as OABC. Several cost function contours are shown there. The optimum solution is at the point B (6, 3). The given point (4, 4) is identified as D on the line B–C in Fig. 13-2.

The gradients of cost and constraint functions at the point D (4, 4) are calculated as

$$\begin{aligned} \mathbf{c} &= (2x_1 - 3x_2 - 10, -3x_1 + 9x_2 - 6) \\ &= (-14, 18) \end{aligned}$$

$$\mathbf{a}^{(1)} = \left(\frac{1}{3}, -\frac{1}{3}\right)$$

$$\mathbf{a}^{(2)} = \left(\frac{1}{12}, \frac{1}{6}\right)$$

$$\mathbf{a}^{(3)} = (-1, 0)$$

$$\mathbf{a}^{(4)} = (0, -1)$$

These gradients are shown at point D in Fig. 13-2. Each constraint gradient points to the direction in which the constraint function value increases. Using these quantities, the QP subproblem of Eqs. (13.1) to (13.3) is defined as

$$\text{minimize } \bar{f} = (-14d_1 + 18d_2) + 0.5(d_1^2 + d_2^2)$$

subject to

$$\begin{bmatrix} \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{12} & \frac{1}{6} \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 0 \\ 4 \\ 4 \end{bmatrix}$$

The solution for the QP subproblem using KKT conditions or the Simplex method of Section 10.4 is

$$\mathbf{d} = (-0.5, -3.5); \quad \mathbf{u} = (43.5, 0, 0, 0)$$

At the solution, only the first constraint is active having a positive Lagrange multiplier. The direction \mathbf{d} is shown in Fig. 13-2. Since the second constraint is inactive, $\mathbf{a}^{(2)} \cdot \mathbf{d}$ must be negative according to Eq. (13.3) and it is (-0.625). Therefore, direction \mathbf{d}

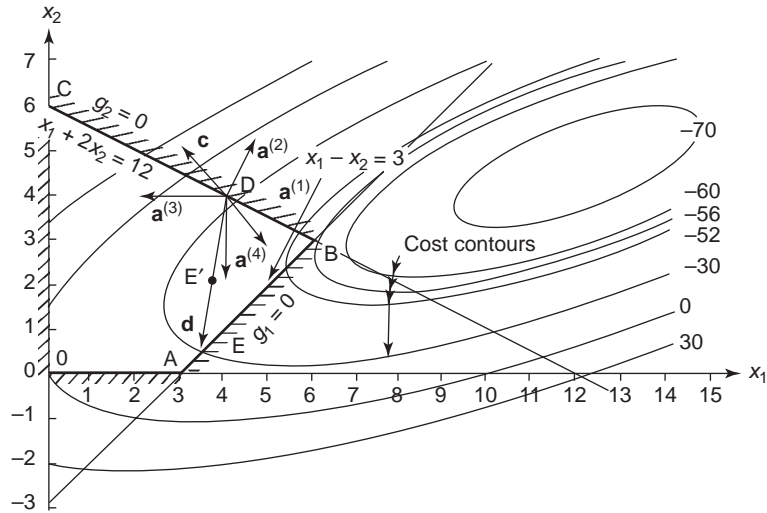


FIGURE 13-2 Feasible region for Example 13.1. Cost reduction step from point D.

points toward the feasible region with respect to the second constraint, which can be observed in Fig. 13-2.

The step size is calculated from Eq. (13.4) based on a 10 percent reduction ($\gamma = 0.1$) of the cost function as

$$\alpha = \frac{|0.1(-24)|}{|(14, -18) \cdot (-0.5, -3.5)|} = 0.3/7$$

Thus, the new design point is given as

$$\mathbf{x}^{(1)} = \begin{bmatrix} 4 \\ 4 \end{bmatrix} + \frac{0.3}{7} \begin{bmatrix} -0.5 \\ -3.5 \end{bmatrix} = \begin{bmatrix} 3.979 \\ 3.850 \end{bmatrix}$$

which is quite close to point D along direction **d**. The cost function at this point is calculated as

$$f(\mathbf{x}^{(1)}) = -26.304$$

which is approximately 10 percent less than the one at the current point (4, 4). It may be checked that all constraints are inactive at the new point.

Direction **d** points into the feasible region at point **D** as can be seen in Fig. 13-2. Any small move along **d** results in a feasible design. If the step size is taken as 1 (which would be obtained if the inaccurate line search of Section 11.3 was performed), then the new point is given as (3.5, 0.5), which is marked as E in Fig. 13-2. At point E, constraint g_1 is active and the cost function has a value of -29.875 , which is smaller than the previous value of -26.304 . If we perform an exact line search, then α is computed as 0.5586 and the new point is given as (3.7207, 2.0449)—identified as point E' in Fig. 13-2. The cost function at this point is -39.641 , which is still better than the one with step size as unity.

Example 13.2 illustrates the cost reduction step with potential constraints.

EXAMPLE 13.2 Cost Reduction Step with Potential Constraints

For Example 13.1, calculate the cost reduction step by considering the potential inequality constraints only.

Solution. In some algorithms, only the potential inequality constraints at the current point are considered while defining the direction finding subproblem, as discussed previously in Section 11.1. The direction determined with this subproblem can be different from that obtained by including all constraints in the subproblem.

For the present problem, only the second constraint is active ($g_2 = 0$) at the point (4, 4). The QP subproblem with this active constraint is defined as

$$\text{minimize } \bar{f} = (-14d_1 + 18d_2) + 0.5(d_1^2 + d_2^2)$$

$$\text{subject to } \frac{1}{12}d_1 + \frac{1}{6}d_2 \leq 0$$

Solving the problem by KKT optimality conditions, we get

$$\mathbf{d} = (14, -18); \quad u = 0$$

Since the Lagrange multiplier for the constraint is zero, it is not active, so $\mathbf{d} = -\mathbf{c}$ is the solution to the subproblem. This search direction points into the feasible region along the negative cost function gradient direction, as seen in Fig. 13-2. An appropriate step size can be calculated along the direction.

If we require the constraint to remain active (i.e., $d_1/12 + d_2/6 = 0$), then the solution to the subproblem is given as

$$\mathbf{d} = (18.4, -9.2); \quad u = -52.8$$

This direction is tangent to the constraint, i.e., along the line D-B in Fig. 13-2.

13.2.2 Constraint Correction Algorithm

If constraint violations are very large at a design point, it may be useful to find out if a feasible design can be obtained. Several algorithms can be used to correct constraint violations. We shall describe a procedure that is a minor variation of the constrained steepest descent method of Section 10.5. A QP subproblem that gives constraint correction can be obtained from Eqs. (10.25) and (10.26) by neglecting the term related to the cost function. In other words, we do not put any restriction on the changes in the cost function, and define the QP subproblem as

$$\text{minimize } \bar{f} = 0.5\mathbf{d}^T\mathbf{d} \tag{13.5}$$

$$\text{subject to } \mathbf{N}^T\mathbf{d} = \mathbf{e} \tag{13.6}$$

$$\mathbf{A}^T\mathbf{d} \leq \mathbf{b} \tag{13.7}$$

A solution to the subproblem gives a direction with the shortest distance to the constraint boundary (linear approximation) from an infeasible point. Equation (13.5) essentially says: *find a direction \mathbf{d} having the shortest path to the linearized feasible region from the current point*. Equations (13.6) and (13.7) impose the requirement of constraint corrections. Note that the potential set strategy as described in Section 11.1 can also be used here. After the direction has been found, a step size can be determined to make sure that the constraint violations are improved. We shall call this procedure the *constraint correction (CC) algorithm*.

Note that constraint correction usually results in an increase in cost. However, there can be some unusual cases where constraint correction is also accompanied by a reduction in the cost function. The constraint correction step is illustrated in Example 13.3.

EXAMPLE 13.3 Constraint Correction Step

For Example 13.1, calculate the constraint correction step from the infeasible point (9, 3).

Solution. The feasible region for the problem and the starting point (F) are shown in Fig. 13-3. The constraint and cost gradients are also shown there. At the point F (9, 3), the following data are calculated:

$$f(9, 3) = -67.5$$

$$g_1 = 1 > 0 \text{ (violation)}$$

$$g_2 = 0.25 > 0 \text{ (violation)}$$

$$g_3 = -9 < 0 \text{ (inactive)}$$

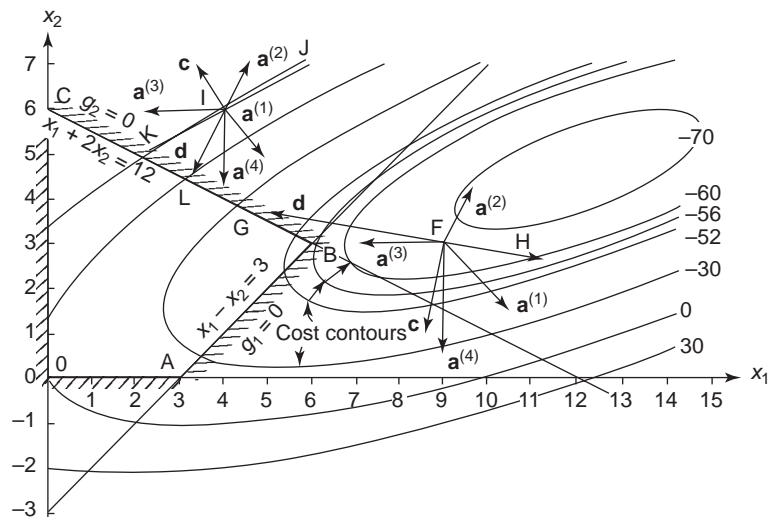


FIGURE 13-3 Feasible region for Example 13.3. Constraint correction and constant cost steps from point F; constant cost step from point I.

$$g_4 = -3 < 0 \text{ (inactive)}$$

$$\mathbf{c} = (-1, -6)$$

and gradients of the constraints are the same as in Example 13.1. Cost and constraint gradients are shown at point F in Fig. 13-3. Thus, the constraint correction QP subproblem of Eqs. (13.5) to (13.7) is defined as

$$\text{minimize } \bar{f} = 0.5(d_1^2 + d_2^2)$$

subject to

$$\begin{bmatrix} \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{12} & \frac{1}{6} \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \leq \begin{bmatrix} -1 \\ -0.25 \\ 9 \\ 3 \end{bmatrix}$$

Using the KKT necessary conditions, the solution for the QP subproblem is given as

$$\mathbf{d} = (-3, 0); \quad \mathbf{u} = (6, 12, 0, 0)$$

Note that the shortest path from Point F to the feasible region is along the line F-B, and the QP subproblem actually gives this solution. The new design point is given as

$$\mathbf{x}^{(1)} = \begin{bmatrix} 9 \\ 3 \end{bmatrix} + \begin{bmatrix} -3 \\ 0 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \end{bmatrix}$$

which is point B in Fig. 13-3. At the new point, constraints g_1 and g_2 are active, and g_3 and g_4 are inactive. Thus, a single step corrects both violations precisely. *This is due to the linearity of all the constraints in the present example.* In general several iterations may be needed to correct the constraint violations. Note that the new point actually represents the optimum solution.

13.2.3 Algorithm for Constraint Correction at Constant Cost

In some instances, the constraint violations are not very large. It is useful to know whether a feasible design can be obtained without any increase in the cost. This shall be called a constant cost subproblem, which can be defined by adding another constraint to the QP subproblem given in Eqs. (13.5) to (13.7). The additional constraint simply requires the current linearized cost to either remain constant or decrease; that is, the linearized change in cost ($\mathbf{c} \cdot \mathbf{d}$) be nonpositive, which is expressed as

$$\mathbf{c} \cdot \mathbf{d} \leq 0 \tag{13.8}$$

The constraint imposes the condition that the direction \mathbf{d} be either orthogonal to the gradient of the cost function (i.e., $\mathbf{c} \cdot \mathbf{d} = 0$), or make an angle between 90° and 270° with it (i.e., $\mathbf{c} \cdot \mathbf{d} < 0$). We shall see this in the example problem discussed later.

If Inequality (13.8) is active (i.e., the dot product is zero, so \mathbf{d} is orthogonal to \mathbf{c}), then there is no change in the linearized cost function value. However, there may be some change in the original cost function due to nonlinearities. If the constraint is inactive, then there is actually some reduction in the linearized cost function along with correction of the constraints. This is a desirable situation. Thus, we observe that a constant cost problem is also a QP subproblem defined in Eqs. (13.5) to (13.8). *It seeks a shortest path to the feasible region that either reduces the linearized cost function or keeps it unchanged.* We shall call this procedure the *correction at constant cost (CCC) algorithm* that is illustrated in Examples 13.4 and 13.5.

Note that the constant cost QP subproblem can be infeasible if the current cost function contour does not intersect the feasible region. This can happen in practice, so a QP subproblem should be solved properly. If it turns out to be infeasible, then the constraint of Eq. (13.8) must be relaxed, and the linearized cost function must be allowed to increase to obtain a feasible point. This will be discussed in the next subsection.

EXAMPLE 13.4 Constraint Correction at Constant Cost

For Example 13.3, calculate the constant cost step from the infeasible point (9, 3).

Solution. To obtain the constant cost step from point F in Fig. 13-3, we impose an additional constraint of Eq. (13.8) as $(\mathbf{c} \cdot \mathbf{d}) \leq 0$ on the QP subproblem given in Example 13.3. Substituting for \mathbf{c} , the constraint is given as

$$-d_1 - 6d_2 \leq 0 \quad (13.9)$$

which imposes the condition that the linearized cost function either remain constant at -67.5 or decrease further. From the graphical representation for the problem in Fig. 13-3, we observe that the cost function value of -67.5 at the given point (9, 3) is below the optimum cost of -55.5 . Therefore, the current cost function value represents a lower bound on the optimum cost function value. However, the linearized cost function line, shown as G–H in Fig. 13-3, intersects the feasible region. Thus, the QP subproblem of Example 13.3 with the preceding additional constraint has feasible solutions. The inequality of Eq. (13.8) imposes the condition that direction \mathbf{d} be either on the line G–H (if the constraint is active) or above it (if the constraint is inactive). In case it is inactive, the angle between \mathbf{c} and \mathbf{d} will be between 90 and 270° . If it is below the line G–H, it violates Inequality (13.8). Note that the shortest path from F to the feasible region is along the line F–B. But this path is below the line G–H and thus not feasible for the preceding QP subproblem.

Solving the problem using KKT conditions, we obtain the solution for the preceding QP subproblem as

$$\mathbf{d} = (-4.5, 0.75); \quad \mathbf{u} = (0, 83.25, 0, 0, 2.4375)$$

Thus the new point is given as

$$\mathbf{x} = (4.5, 3.75) \quad \text{with} \quad f = -34.6$$

At the new point (G in Fig. 13-3), all the constraints are inactive except the second one (g_2). The constant cost condition of Eq. (13.8) is also active, which implies that

the direction \mathbf{d} is orthogonal to the cost gradient vector \mathbf{c} . As seen in Fig. 13-3, this is indeed true. Note that because of the highly nonlinear nature of the cost function at the point F (9, 3), the new cost function actually increases substantially. Thus the direction \mathbf{d} is not truly a constant cost direction. Although the new point corrects all the violations, the cost function increases beyond the optimum point, which is undesirable. Actually, from point F it is better to solve just the constraint correction problem, as in Example 13.3. The increase in cost is smaller for that direction. Thus, in certain cases, it is better to solve just the constraint correction subproblem.

EXAMPLE 13.5 Constraint Correction at Constant Cost

Consider another starting point as (4, 6) for Example 13.3, and calculate the constant cost step from there.

Solution. The starting point is identified as I in Fig. 13-3. The following data are calculated at the point (4, 6):

$$f(4, 6) = 30$$

$$g_1 = -\frac{5}{3} \text{ (inactive)}$$

$$g_2 = \frac{1}{3} > 0 \text{ (violated)}$$

$$g_3 = -4 < 0 \text{ (inactive)}$$

$$g_4 = -6 < 0 \text{ (inactive)}$$

$$\mathbf{c} = (-20, 36)$$

and the constraint gradients are the same as in Example 13.1. Cost and constraint gradients are shown in Fig. 13-3 at point I. Note that the cost function at point I is above the optimum value. Therefore, the constant cost constraint of Eq. (13.8) may not be active for the solution of the subproblem, i.e., we may be able to correct constraints and reduce the cost function at the same time.

The constant cost QP subproblem given in Eqs. (13.5) to (13.8) is defined as

$$\text{minimize } \bar{f} = 0.5(d_1^2 + d_2^2)$$

subject to

$$\begin{bmatrix} \frac{1}{3} & -\frac{1}{3} \\ \frac{1}{12} & \frac{1}{6} \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \leq \begin{bmatrix} \frac{5}{3} \\ -\frac{1}{3} \\ 4 \\ 6 \end{bmatrix}$$

$$-20d_1 + 36d_2 \leq 0$$

Writing the KKT conditions for the QP subproblem, we obtain the solution as

$$\mathbf{d} = (-0.8, -1.6); \quad \mathbf{u} = (0, 9.6, 0, 0, 0)$$

Note that only the second constraint is active. Therefore, the new point should be precisely on the constraint equation $x_1 + 2x_2 = 12$, shown as point L in Fig. 13-3. The constant cost constraint is not active (the direction \mathbf{d} lies below the line J–K in Fig. 13-3 making an angle greater than 90° with \mathbf{c}). Thus the new cost function value should decrease at the new point (3.2, 4.4), and it does; -3.28 versus 30, which is a substantial reduction.

13.2.4 Algorithm for Constraint Correction at Specified Increase in Cost

As observed in the previous subsection, the constant cost subproblem can be infeasible. In that case, the current value of the cost function must be allowed to increase. This can be done quite easily by slightly modifying Inequality (13.8) as

$$\mathbf{c}^T \mathbf{d} \leq \Delta \quad (13.10)$$

where Δ is a specified limit on the increase in cost. The increase in cost can be specified based on the condition that the new cost based on the linearized expression does not go beyond the previous cost at a feasible point, if known. Note again that the QP subproblem in this case can be infeasible if the increase in the cost specified in Δ is not enough. Therefore, Δ may have to be adjusted. We shall call this procedure the *constraint correction at specified cost (CCS) algorithm* that is illustrated in Example 13.6.

EXAMPLE 13.6 Constraint Correction at Specified Increase in Cost

For Example 13.4, calculate the constraint correction step from the infeasible point (9, 3) with a 10 percent increase in cost.

Solution. Since the current value of the cost function is -67.5 , the 10 percent increase in cost gives Δ as 6.75 in Eq. (13.10). Therefore, using $\mathbf{c} = (-1, -6)$ as calculated in Example 13.3, the constraint of Eq. (13.10) becomes

$$-d_1 - 6d_2 \leq 6.75$$

Other constraints and the cost function are the same as defined in Example 13.3. Solving the problem using KKT conditions, we obtain the solution of the subproblem as

$$\mathbf{d} = (-3, 0); \quad \mathbf{u} = (6, 12, 0, 0, 0)$$

At the solution, the first two constraints are active, and the constraint of Eq. (13.10) is inactive. Note that this is the same solution as obtained in Example 13.3. Thus the new point represents the optimum solution.

13.2.5 Constraint Correction with Minimum Increase in Cost

It is possible to define a subproblem that minimizes the increase in cost and at the same time corrects constraints. The subproblem is defined as

$$\text{minimize } f_L = \mathbf{c}^T \mathbf{d}$$

subject to the constraints of Eqs. (13.6) and (13.7), where f_L is the linearized change in the cost function. This problem may be unbounded, so we impose the following move limits

$$-\Delta_i \leq d_i \leq \Delta_i; \quad i = 1 \text{ to } n$$

to obtain a bounded problem. Here Δ_i represents the maximum and minimum value for d_i . The preceding subproblem is linear, so any LP code can be used to solve it. A line search can be performed along the direction \mathbf{d} to find the proper step size. Example 13.7 illustrates calculations for constraint correction with minimum increase in cost.

EXAMPLE 13.7 Constraint Correction with Minimum Increase in Cost

For Example 13.4, solve the constraint correction problem with the minimum increase in cost from point F (9, 3) shown in Fig. 13-3.

Solution. At the point (9, 3) the following data are calculated in Example 13.3:

$$f(9, 3) = -67.5$$

$$g_1 = 1 > 0 \text{ (violation);}$$

$$g_2 = 0.25 > 0 \text{ (violation);}$$

$$g_3 = -9 < 0 \text{ (inactive)}$$

$$g_4 = -3 < 0 \text{ (inactive)}$$

$$\mathbf{c} = (-1, -6).$$

Therefore, the subproblem is defined as

$$\text{minimize } f_L = -d_1 - 6d_2$$

subject to

$$\frac{1}{3}d_1 - \frac{1}{3}d_2 \leq -1$$

$$\frac{1}{12}d_1 + \frac{1}{6}d_2 \leq -0.25$$

$$-d_1 \leq 9$$

$$-d_2 \leq 3$$

$$-\Delta_1 \leq d_1 \leq \Delta_1; \quad -\Delta_2 \leq d_2 \leq \Delta_2$$

The linear programming subproblem can be solved using the Simplex method used in Chapter 6. We solve the problem using the program LINDO (Schrage, 1981). With $\Delta_1 = \Delta_2 = 1$, the problem is infeasible, the move limits are too restrictive, and the feasible point cannot be found. Since the problem has two variables, one can easily graph all the problem functions and verify that there is no solution to the preceding linearized subproblem. When $\Delta_1 = \Delta_2 = 3$, the following solution is obtained:

$$d_1 = -3, \quad d_2 = 0, \quad f_L = 3$$

and the second constraint is active with the Lagrange multiplier as 36. The lower limit on d_1 is also active with the Lagrange multiplier as 2.0. When d_1 and d_2 are added to the starting point (9, 3), the new point is given as (6, 3). This is actually the optimum point with the cost function value as -55.5 . Note that since $f_L = 3$, the cost function was supposed to increase by only 3 from -67.5 . However, because of nonlinearity, it has actually increased by 12.

Note that since the Lagrange multiplier for the lower bound constraint on d_1 is 2, the Constraint Sensitivity Theorem 4.7 predicts that f_L will decrease by 2 to 1 if Δ_1 is changed to 4. This is indeed the case. With $\Delta_1 = \Delta_2 = 4$, the solution of the subproblem is obtained as

$$d_1 = -4, \quad d_2 = 0.5, \quad f_L = 1.0$$

and the second constraint is active with the Lagrange multiplier as 36. The lower limit on d_1 is still active with the Lagrange multiplier as 2.0. The new point is given as (5, 3.5) with the cost function as -43.375 . For this point the cost function is actually increased by 24.125 rather than just 1 as predicted by the solution of the linearized subproblem.

13.2.6 Observations on Interactive Algorithms

We have discussed several algorithms that are useful for interactive design optimization. They are demonstrated for a problem that has linear constraints and quadratic cost function. There are certain limitations of these algorithms that should be clearly understood:

1. *All the algorithms use linear approximations for the cost and constraint functions.* For highly nonlinear problems, the solution of the subproblems are therefore valid for a small region around the current point.
2. *The step size calculated in Eq. (13.4) using the desired reduction γ in the cost function is based on the linear approximation for the cost function.* With the calculated step size, the *actual reduction* in the cost function may be *smaller* or *larger* than γ depending on the nonlinearity of the cost function.
3. *In the constraint correction problem of Example 13.3, only one step is needed to correct all the constraints.* This is due to the linearity of all the constraints. When the constraints are nonlinear, *several constraint correction steps are usually needed* to reach the feasible region. The spring design problem solved later in Section 13.5 demonstrates this fact.
4. *Constraint correction is most often accompanied by an increase in the cost function.* However, in certain cases it may also result in a decrease in the cost function. This is rare and depends on the nonlinearity of functions and the starting point.

5. The *constant cost condition* of Eq. (13.8) is based on the linearized cost function. Even if this constraint is active at the solution for the subproblem, *there may be changes to the original cost function at the new point*. This is due to the *nonlinearity* of the cost and constraint functions. We have observed this phenomena in Examples 13.4–13.7.
6. For some infeasible points, *it is better to solve the constraint correction subproblem rather than the constant cost subproblem*.
7. As seen in Examples 13.1 and 13.2, *there are several cost reduction directions at a given feasible point*. They depend on the definition of the QP subproblem. It is difficult to determine the best possible direction.
8. *The Lagrange multipliers evaluated during the solution of QP subproblems can be quite different* from their values at the optimum solution to the original problem. This can be observed in the solution of Examples 13.1–13.7.

13.3 Desired Interactive Capabilities

Interactive software for design optimization should be flexible and user-friendly. Help facilities should be available in the program which should have graphical user interface or be menu driven. We shall mention several desirable capabilities of such interactive software.

The program should be able to treat general nonlinear programming as well as unconstrained problems; treat equality, inequality, and design variable bound constraints; should have choice of a few good algorithms that are robustly implemented; and trap users' mistakes and not abort abnormally.

13.3.1 Interactive Data Preparation

The software should have a module for interactive data preparation and editing. The commands for data entry should be explicit. Only *the minimum amount of data* should be required. The user should be able to edit any data that have been entered previously. The step-by-step procedure should be set up to display the *menu* for data selection and entry. Or, it should be possible to enter data in a simple *question/answer* session. The system should be set up in such a way so that it is *protected* from any of the designer's mistakes. If data mismatch is found, *messages* should be given in detail. The interactive input procedure should be *simple* so that even a beginner can follow it easily.

13.3.2 Interactive Capabilities

As observed earlier, it is prudent to allow designer interaction in the computer-aided design process. Such a dialog can be very beneficial, saving computer and human resources. For the use of the interactive software system in engineering, two questions arise: (1) what are the *advantages and disadvantages* of the interaction, and (2) what *type of interactive capability* needs to be provided? We shall address both these questions.

All general-purpose design optimization software need the following information about the problem to be solved: (1) input data such as number of design variables, number of constraints, and so on, (2) the cost and constraint functions, and (3) the gradients of cost and constraint functions. If the gradients are not available, then the system should automatically approximate them by a finite difference method. If there is a mistake in the input data or problem definition, errors will occur in the problem-solving procedure. The optimization system should take care of such mistakes as much as possible.

It is also useful to monitor the optimum process through the interactive session. *Histories of the cost function, constraint functions, design variables, maximum constraint violation, and convergence parameter should be monitored. When these histories are graphically dis-*

played, they can be of great help in certain cases of decision making. If the design process is not proceeding satisfactorily (there could be inaccuracies or errors in the problem formulation and modeling), it is necessary to terminate it and check the formulation of the problem. This will save human as well as computer resources. Also, if one algorithm is not progressing satisfactorily, a switch should be made to another one. The system should be able to give suggestions for design change based on the analysis of the trends. Therefore, monitoring the iterative process interactively is an important capability that should be available in design optimization software.

The designer should also be able to guide the problem-solving process. For example, the program can be run for a certain number of iterations and interrupted to see if the process is progressing satisfactorily. If it is not progressing as expected, *a decision to change the course of calculations can be made.* If there are constraint violations, the designer may want to know whether they can be *corrected without any penalty* to the cost function. If this cannot be done, the *penalty* to the cost function to correct the constraints should be made available. When the design is in the feasible region, the system should have the capability to perform calculations and determine if the cost function can be reduced by a certain percentage and still remain feasible. *If the iterative process does not progress well, then the designer should be able to restart the program from any previous iteration or any other design.* At the optimum point, the penalty to tighten a constraint or the gain to relax it should be displayed. This information is available from the Lagrange multipliers for the constraints. In practical optimization, these interactive capabilities can be quite useful.

It should be possible to change the input data for a design problem during the iterative process. After monitoring the process for a few iterations, it may be necessary to *change the problem or program parameters.* This should be possible without terminating the program. Design sensitivity coefficients of the cost function and potential constraints should be displayed in a convenient form, e.g., as normalized bar charts. This information will show *relative sensitivity* of the design variables. The designer should also be able to determine the *status of the design variables* and change it interactively if desired. *The trend information* when displayed graphically can aid the designer in gaining insights into the problem behavior so this capability should be available.

It should also be possible to utilize the interactive design optimization software in the batch environment with a minimum of input data. The system should have default values for the best parameters determined through expertise and numerical experimentation.

13.3.3 Interactive Decision Making

When the program is run interactively, a wide range of options should be available to the designer. The following is a list of possible capabilities that can aid the designer in decision making:

1. The designer may want to *re-examine the problem formulation* or design data. Therefore, it should be possible to terminate the program properly and restart it.
2. It should be possible at any iteration to display the *status of the design*, such as current values of variables, cost function, maximum constraint violation, and other such data.
3. It should be possible to *change data* at any iteration, such as design variables and their limits, convergence criteria, and other data.
4. The designer should be able to *fix design variables* to any value. It should also be possible to release the fixed design variables.
5. The designer should be able to *run the algorithm one iteration* at a time or several iterations.
6. It should be possible to *restart* the program from any iteration.

7. It should be possible to *change the algorithm* during the iterative process.
8. The designer should be able to *request a reduction in the cost function* by x percent from a feasible point.
9. The designer should be able to *request a constraint correction* at any iteration.
10. The designer should be able to *request a constant cost step*.
11. The designer should be able to *request a constraint correction* with an x percent limit on the increase in cost.
12. The designer should be able to *request various graphical displays*.

13.3.4 Interactive Graphics

Graphical display of data is a convenient way to interpret results and draw conclusions. Interactive graphics can play a major role in design decision making during the iterative optimization process. Possible graphical displays are:

1. Plots of cost function, convergence parameters, and maximum constraint violation *histories*. These show the progress of the iterative process toward the optimum point.
2. *Histories of design variables*. These can be used to observe the trend in design variables and possibly used to extrapolate their values.
3. *Constraint function histories* can be displayed. This can show constraints that are not playing any role in the design process. It can also show dominant constraints.
4. *Sensitivity coefficients* for the cost and constraint functions can be displayed in the form of bar charts. These are nothing but normalized gradients of cost and constraint functions showing sensitive or insensitive variables and functions.

It can be seen that by using interactive graphics capabilities, designers can observe the progress of the optimization process. They can *learn* more about the behavior of the design problem and perhaps *refine* its formulation.

13.4 Interactive Design Optimization Software

The preceding sections essentially describe specifications for a general-purpose interactive design optimization software. Based on them, a software system can be designed and implemented. It can be observed that to implement all the flexibilities and capabilities, the software will be quite large and complex. The most modern software design and data management techniques will have to be utilized to achieve the stated goals. The entire process of software design, implementation, and evaluation can be quite costly and time-consuming, requiring the equivalent of several man-years.

In this section, we shall briefly describe software that has some of the previously stated capabilities. Other available software may also have similar capabilities. The present program is called IDESIGN, which stands for *Interactive Design Optimization of Engineering Systems*. It has interactive and graphics facilities suitable for computer-aided optimization and design (Arora, 1984; Arora and Tseng, 1987a,b). With the IDESIGN program, the computer and the designer's experience can be utilized to adjust design variables so as to improve the design objective while satisfying the constraints. It contains four state-of-the-art nonlinear optimization algorithms. Efficient and reliable implementations of the algorithms have been developed over several years of testing. The simpler cases of linear and unconstrained optimization can also be handled.

IDESIGN has several facilities that permit the engineer to interact with and control the optimization process. The designer can backtrack to any previous design or manually input

a new trial design. Design information can be displayed in a variety of ways or represented in graphs. The system has been designed to accommodate both experienced users and beginners. The beginner can respond to one menu at a time as guided by online instruction, whereas the expert can prepare an input data file and thus bypass immediate menus. The software identifies and helps the user correct improper responses. Input and output can be echoed to a “dialog” file for the user’s reference. Input can also be received from a file for batch mode operation for large-scale problems.

13.4.1 User Interface for IDESIGN

IDESIGN consists of a main program and several standard subroutines that need not be changed by the user. In order to solve a design problem, the user must prepare additional subroutines for the program. The input data, such as the initial design, lower and upper limits on design variables, problem parameters, and the parameter values to invoke various options available in the program, must also be provided. The input data and options available in the program are described in the user’s manual (Arora and Tseng, 1987a).

The user must describe the design problem by coding the following four FORTRAN subroutines:

- USERMF: Minimization (cost) Function evaluation subroutine
- USERCF: Constraint Functions evaluation subroutine
- USERMG: Minimization (cost) function Gradient evaluation subroutine
- USERCG: Constraint functions Gradient evaluation subroutine

A fifth subroutine USEROU may also be provided by the user to perform postoptimality analyses for the optimum solution and obtain more output using specialized formats.

Figure 13-4 shows a conceptual layout of the interactive design optimization environment with the program IDESIGN. To create a design system for a particular application, the

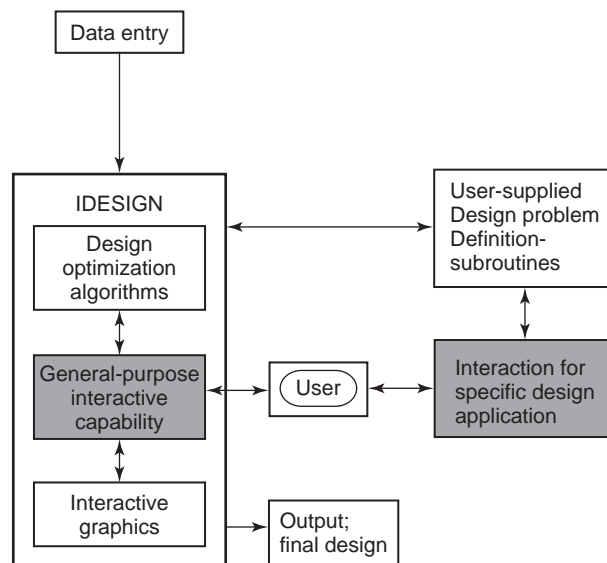


FIGURE 13-4 Conceptual layout of interactive design optimization environment with IDESIGN.

designer needs to develop FORTRAN subroutines that define the problem—cost and constraint functions as well as gradient evaluation subroutines. The designer has all the flexibilities to develop these subroutines as long as the “argument” requirements to interface with IDESIGN are satisfied. For example, additional arrays may be declared, external subroutines or independent programs may be called, and additional input data may be entered. Through these subroutines, the designer may also incorporate more interactive commands that are specific to the domain of the application.

General-purpose interactive capability is available in IDESIGN as shown on the left side of Fig. 13-4. In this part, interactive commands that are not connected to any specific area of application are available. Table 13-1 contains a list of commands that are currently available. Using these commands, the designer can interactively guide the process toward acceptable designs. The command CH/XXX is particularly useful, as it allows the designer to change design variable values and their upper and lower limits, algorithm, and convergence criteria. It can also be used for obtaining advice from IDESIGN for the best changes to design variables to correct constraints. The PLOT commands can be used to observe trends in the design variables, determine critical constraints, and determine sensitive and insensitive variables with respect to the cost and constraint functions.

It can be seen that the foregoing interactive facilities can be utilized to gain insights into the behavior of a particular design problem. Having gained this knowledge, the designer can perhaps develop alternate design concepts that may be more efficient and economical.

TABLE 13-1 Interactive Commands Available in IDESIGN

<i>Command</i>	<i>Purpose</i>
CON	CONTINUE IDESIGN
DIS	DISPLAY THE DATA
HELP	HELP THE USER
QUIT	STOP IDESIGN
PLOT/NO.	NO = 1 COST HISTORY NO = 2 CONVERGENCE PARAMETER HISTORY NO = 3 MAX CONSTRAINT VIOLATION HISTORY NO = 4 DESIGN VARIABLES HISTORY NO = 5 CONSTRAINTS HISTORY NO = 6 CONSTRAINT AND COST SENSITIVITY BAR CHARTS
CH/XXX	CHANGE VARIABLES OR PARAMETERS XXX = ABBREVIATION OF PARAMETER
OPT	GO TO OPTIMUM
OK/XX	READY TO CONTINUE IDESIGN XX = NUMBER OF ITERATIONS; IF/XX IS OMITTED, IDESIGN GOES TO NEXT DESIGN POINT FOR OK/FEA, IDESIGN GOES TO NEXT FEASIBLE DESIGN
RS/XXX	RESTART FROM ITERATION NUMBER XXX
CR	SOLVES COST REDUCTION PROBLEM
CC	SOLVES CONSTRAINT CORRECTION PROBLEM
CCC	SOLVES CONSTRAINT CORRECTION AT CONSTANT COST PROBLEM
CCS	SOLVES CONSTRAINT CORRECTION WITH BOUND ON INCREASE IN COST

13.4.2 Capabilities of IDESIGN

The program has been used to solve several classes of optimal design problems:

1. *Small-scale engineering design problems* having explicit cost and constraint functions, such as the ones described earlier in this text.
2. *Structural design problems* modeled using finite elements, such as trusses, frames, mixed finite elements, bridges, industrial buildings, high-rise buildings, plate girders, machine elements, and many others (Arora and Haug, 1979; Arora and Thanedar, 1986; Arora, 2002). More details of applications in this area are also given in Chapter 14.
3. *Dynamic response optimization* applications, such as vibration isolation, steady-state response, designs for earthquake resistance, worst-case design, and transient response problems (Hsieh and Arora, 1984; Lim and Arora, 1987; Tseng and Arora, 1987; Arora, 1999).
4. *Biomechanics applications*, such as muscle force distribution and contact force determination problems (Pederson *et al.*, 1987).
5. *Optimal control of systems*—structural, mechanical, and aerospace applications. More details of applications in this area are discussed in Chapter 14.
6. *System identification problems*, such as environmental and material modeling problems (Kim and Arora, 2003).

Problem Type and Algorithms The program can solve any general nonlinear programming problem formulated as given in Eq. (10.1), linear programming problems, and unconstrained problems. Although the program has the option of solving linear programming problems, the algorithm used is not as efficient as the Simplex method. So, for large linear programming problems, it is suggested that a program based on the Simplex method be used. The following *algorithms* are available:

1. *Cost function bounding* algorithm (Arora, 1984a,b).
2. *Pshenichny's linearization method* (Section 10.5; Belegundu and Arora, 1984).
3. *Sequential quadratic programming* algorithm that generates and uses approximate second-order information for the Lagrange function (Section 11.4; Lim and Arora, 1986).
4. A *hybrid method* that combines the cost function bounding and the sequential quadratic programming algorithms (Thanedar *et al.*, 1986).
5. *Conjugate gradient* method for unconstrained problems (Section 8.4).

If an algorithm is not specified by the user, the program automatically uses the best algorithm.

Gradient Evaluation The following capabilities to evaluate gradients and check gradient expressions are available:

1. If the user does not program gradient expressions in USERMG and USERCG subroutines, the program has an option to *automatically calculate* them.
2. An option is available in IDESIGN to determine the *optimum value of δ* for the finite difference gradient evaluation of cost and constraint functions.
3. If the user has programmed *gradient expressions* in USERMG and USERCG subroutines, an option is available to *verify* them, i.e., the gradient evaluation is checked using the finite difference approach. If the gradient expressions are in error, an option is available to either stop the program or continue its execution.

These options have proved to be useful in practical applications.

Output Several levels of output can be obtained from the program. This can be specified in the input data. The minimum output giving the final design, design variables and constraint activities, and histories of cost function, convergence parameter and maximum constraint violation, can be obtained. More detailed information at each iteration, such as the gradient matrix and other intermediate results, can also be obtained.

13.5 Examples of Interactive Design Optimization

In this section, we shall demonstrate the use of some of the interactive capabilities by solving the spring design optimization problem formulated in Section 2.9 (Shigley and Mischke, 2001). Given numerical data will be used to solve the problem using batch and interactive capabilities.

13.5.1 Formulation of the Spring Design Problem

Standard Definition of the Problem After *normalizing* the constraints, using the defined data and writing them in the standard form of Section 2.11, we obtain the optimum design formulation for the spring problem as

$$\text{minimize } f = (N+2)Dd^2 \quad (13.11)$$

subject to the deflection constraint

$$g_1 = 1.0 - \frac{D^3 N}{(71875d^4)} \leq 0 \quad (13.12)$$

the shear stress constraint

$$g_2 = \frac{D(4D-d)}{12566d^3(D-d)} + \frac{2.46}{12566d^2} - 1.0 \leq 0 \quad (13.13)$$

the surge wave frequency constraint

$$g_3 = 1.0 - \frac{140.54d}{D^2 N} \leq 0 \quad (13.14)$$

and the outer diameter constraint

$$g_4 = \frac{D+d}{1.5} - 1.0 \leq 0 \quad (13.15)$$

The lower and upper bounds on the design variables are selected as follows:

$$\begin{aligned} 0.05 &\leq d \leq 0.20 \text{ in} \\ 0.25 &\leq D \leq 1.30 \text{ in} \\ 2 &\leq N \leq 15 \end{aligned} \quad (13.16)$$

Note that the constant $\pi^2\rho/4$ in the cost function of Eq. (13.11) has been neglected. This simply scales the cost function value without affecting the final optimum solution. The problem has three design variables and 10 inequality constraints in Eqs. (13.12) to (13.16). If we attempt to solve the problem analytically using the KKT conditions of Section 4.4, we will have to consider 2^{10} cases, which will be tedious and time-consuming.

13.5.2 Optimum Solution for the Spring Design Problem

Any suitable program can be used to solve the problem defined in Eqs. (13.11) to (13.16). We solve the problem using the sequential quadratic programming algorithm of Section 11.4 available in the IDESIGN software package. The history of the iterative design process is shown in Table 13-2. The table shows iteration number (Iter.), maximum constraint violation (Max. vio.), convergence parameter (Conv. parm.), cost function (Cost), and design variable values at each iteration. It also gives constraint activity at the optimum point indicating whether a constraint is active or not, constraint function values, and their Lagrange multipliers. Design variable activity is shown at the optimum point, and the final cost function value and the number of calls to user routines are also given.

The following stopping criteria are used for the present problem:

1. The maximum constraint violation (Max. vio.) should be less than ε_1 , i.e., $\mathbf{V} \cong \varepsilon_1$ in Step 4 of the algorithm given in Section 11.4. ε_1 is taken as 1.00E-04.
2. The length of the direction vector (Conv. parm.) should be less than ε_2 , i.e., $\|\mathbf{d}\| \cong \varepsilon_2$ in Step 4 of the algorithm given in Section 11.4. ε_2 is taken as 1.00E-03.

The starting design estimate is (0.2, 1.3, 2.0), where the maximum constraint violation is 96.2 percent and the cost function value is 0.208. At the sixth iteration, a feasible design (maximum constraint violation is 1.97E-05) is obtained at a cost function value of (1.76475E-02). Note that in this example, the constraint correction is accompanied by a substantial reduction (by a factor of 10) in the cost function. However, most often, the constraint correction will result in an increase in cost. The program takes another 12 iterations to reach the optimum design. At the optimum point, the deflection and shear stress constraints of Eqs. (13.13) and (13.14) are active. The Lagrange multiplier values are (1.077E-02) and (2.4405E-02). Design variable one (wire diameter) is close to its lower bound.

13.5.3 Interactive Solution for Spring Design Problem

In the previous subsection, the spring design problem was *solved in the batch environment where the designer had no control over the iterative process*. The program took 18 iterations to converge to the optimum point. We shall solve the problem interactively starting from the same design point. The procedure will be to *interrupt* the program at every iteration, analyze the design conditions, and give *interactive commands* to execute a particular step. In the current application, only the cost function value and maximum constraint violation are monitored and used to make decisions. In more advanced applications, histories of design variables and other graphic facilities can also be used to make decisions. For example, design variable values can be extrapolated based on the observation of trends. This will be demonstrated in the next subsection.

Table 13-3 contains histories of design variables, maximum constraint violation, convergence parameter, and the cost function. It also shows the interactive algorithm used at each iteration. The initial objective is to obtain a feasible design so the constraint correction (CC) algorithm is executed for the first six iterations. A feasible design is obtained at the seventh iteration. Note that during the first six iterations, constraint correction is accompanied by a reduction in the cost function. At the seventh iteration, the cost reduction (CR) algorithm is executed with a 20 percent reduction in the cost function. At the eighth iteration the cost

TABLE 13-2 History of the Iterative Optimization Process for the Spring Design Problem in Batch Environment

<i>Iter.</i>	<i>Max. vio.</i>	<i>Conv. parm</i>	<i>Cost</i>	<i>d</i>	<i>D</i>	<i>N</i>
1	9.61791E-01	1.00000E+00	2.08000E-01	2.0000E-01	1.3000E+00	2.0000E+00
2	2.48814E+00	1.00000E+00	1.30122E-02	5.0000E-02	1.3000E+00	2.0038E+00
3	6.89874E-01	1.00000E+00	1.22613E-02	5.7491E-02	9.2743E-01	2.0000E+00
4	1.60301E-01	1.42246E-01	1.20798E-02	6.2522E-02	7.7256E-01	2.0000E+00
5	1.23963E-02	8.92216E-03	1.72814E-02	6.8435E-02	9.1481E-01	2.0336E+00
6	1.97357E-05	6.47793E-03	1.76475E-02	6.8770E-02	9.2373E-01	2.0396E+00
7	9.25486E-06	3.21448E-02	1.76248E-02	6.8732E-02	9.2208E-01	2.0460E+00
8	2.27139E-04	7.68889E-02	1.75088E-02	6.8542E-02	9.1385E-01	2.0782E-00
9	5.14338E-03	8.80280E-02	1.69469E-02	6.7635E-02	8.7486E-01	2.2346E+00
10	8.79064E-02	8.87076E-02	1.44839E-02	6.3848E-02	7.1706E-01	2.9549E+00
11	9.07017E-02	6.66881E-02	1.31958E-02	6.0328E-02	5.9653E-01	4.0781E+00
12	7.20705E-02	7.90647E-02	1.26517E-02	5.7519E-02	5.1028E-01	5.4942E+00
13	6.74501E-02	6.86892E-02	1.22889E-02	5.4977E-02	4.3814E-01	7.2798E+00
14	2.81792E-02	4.50482E-02	1.24815E-02	5.3497E-02	4.0092E-01	8.8781E+00
15	1.57825E-02	1.94256E-02	1.25465E-02	5.2424E-02	3.7413E-01	1.0202E+01
16	5.85935E-03	4.93063E-03	1.26254E-02	5.1790E-02	3.5896E-01	1.1113E+01
17	1.49687E-04	2.69244E-05	1.26772E-02	5.1698E-02	3.5692E-01	1.1289E+01
18	0.00000E+00	9.76924E-08	1.26787E-02	5.1699E-02	3.5695E-01	1.1289E+01

Constraint activity

<i>No.</i>	<i>Active</i>	<i>Value</i>	<i>Lagr. mult.</i>
1	Yes	-4.66382E-09	1.07717E-02
2	Yes	-2.46286E-09	2.44046E-02
3	No	-4.04792E+00	0.00000E+00
4	No	-7.27568E-01	0.00000E+00

Design variable activity

<i>No.</i>	<i>Active</i>	<i>Design</i>	<i>Lower</i>	<i>Upper</i>	<i>Lagr. mult.</i>
1	Lower	5.16987E-02	5.00000E-02	2.00000E-01	0.00000E+00
2	Lower	3.56950E-01	2.50000E-01	1.30000E+00	0.00000E+00
3	No	1.12895E+01	2.00000E+00	1.50000E+01	0.00000E+00

No. of calls for cost function evaluation (USERMF) = 18

No. of calls for evaluation of cost function gradient (USERMG) = 18

No. of calls for constraint function evaluation (USERCF) = 18

No. of calls for evaluation of constraint function gradients (USERCG) = 18

No. of total gradient evaluations = 34

function is reduced but constraint violation again appears. For the next two iterations, constraint correction at constant cost (CCC) is sought and a nearly feasible design is obtained at the tenth iteration. At the tenth iteration, constraint correction at a specified increase in cost (CCS) is sought. At the eleventh iteration, all constraints are satisfied and the convergence parameter is quite small, so the program is terminated. The cost function is fairly close to the true optimum. However, the design point is somewhat different. It turns out that there are several near optimum designs in the neighborhood of the true optimum for this example problem.

TABLE 13-3 Interactive Solution Process for the Spring Design Problem

<i>Iter.</i>	<i>Algor.</i>	<i>Max. vio.</i>	<i>Conv. parm.</i>	<i>Cost</i>	<i>d</i>	<i>D</i>	<i>N</i>
1	CC	9.61791E-01	1.00000E+00	2.08000E-01	2.0000E-01	1.3000E+00	2.0000E+00
2	CC	2.48814E-00	1.00000E+00	1.30122E-02	5.0000E-02	1.3000E+00	2.0038E+00
3	CC	6.89874E-01	1.00000E+00	1.22613E-02	5.7491E-02	9.2743E-01	2.0000E+00
4	CC	1.60301E-01	1.00000E+00	1.20798E-02	6.2522E-02	7.7256E-01	2.0000E+00
5	CC	3.70554E-01	1.00000E+00	1.03315E-02	5.8477E-02	5.1558E-01	3.8601E+00
6	CC	5.06054E-01	1.00000E+00	7.96802E-03	5.0000E-02	2.9195E-01	8.9170E+00
7	CR	0.00000E+00	1.67623E-02	1.47352E-02	5.5455E-02	4.3230E-01	9.0837E+00
8	CCC	3.53358E-02	1.67623E-02	1.19085E-02	5.2692E-02	3.8896E-01	9.0828E+00
9	CCC	4.24950E-04	1.67623E-02	1.27298E-02	5.3485E-02	4.0151E-01	9.0831E+00
10	CCS	1.08957E-04	1.67623E-02	1.27290E-02	5.3395E-02	3.9916E-01	9.1854E+00
11	CR	0.00000E+00	5.49055E-05	1.27300E-02	5.3396E-02	3.9918E-01	9.1854E+00

Constraint activity

<i>No.</i>	<i>Active</i>	<i>Value</i>	<i>Lagr. mult.</i>
1	Yes	-2.94670E-09	1.09581E-02
2	Yes	-1.36188E-09	2.45745E-02
3	No	-4.12384E+00	0.00000E+00
4	No	-6.98284E-01	0.00000E+00

Design variable activity

<i>No.</i>	<i>Active</i>	<i>Design</i>	<i>Lower</i>	<i>Upper</i>	<i>Lagr. mult.</i>
1	Lower	5.33956E-02	5.00000E-02	2.00000E-01	0.00000E+00
2	No	3.99178E-01	2.50000E-01	1.30000E+00	0.00000E+00
3	No	9.18539E+00	2.00000E+00	1.50000E+01	0.00000E+00

No. of calls for cost function evaluation = 11

No. of calls for evaluation of cost function gradient = 11

No. of calls for constraint function evaluation = 11

No. of calls for evaluation of constraint function gradient = 11

No. of total gradient evaluations = 20

13.5.4 Use of Interactive Graphics

The graphical display of a large amount of data is an extremely convenient way to interpret results and draw conclusions. Interactive graphics can play a major role in decision making during the design process. We demonstrate the possible use of interactive graphics during the design optimization process using the spring design problem as an example. We execute the spring design problem for 10 iterations starting from the point (0.2, 1.3, 2.0). At the tenth iteration the program is interrupted and the execution control is transferred to the designer. Various plotting commands available in the program are used to display the data on the screen. Next, we will explain various graphics facilities and their possible use in the practical design environment.

Design Variable Trend The history of design variables when displayed on the screen can show their trend. For example, Fig. 13-5 shows the variation of design variables as the iterations progress. It shows that design variable 1 decreases at the first iteration and then remains almost constant. If the information were displayed at an intermediate iteration, the variable could be assigned a fixed value since it was not changing very much. Design variable 2

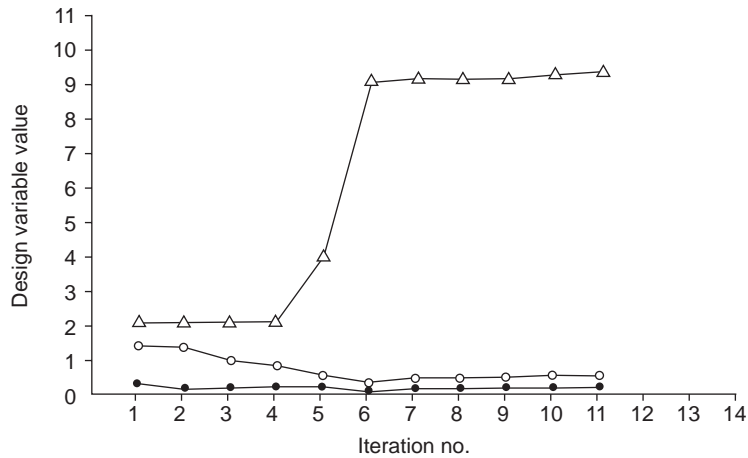


FIGURE 13-5 History of design variables for the spring design problem. •, d ; ○, D ; △, N .

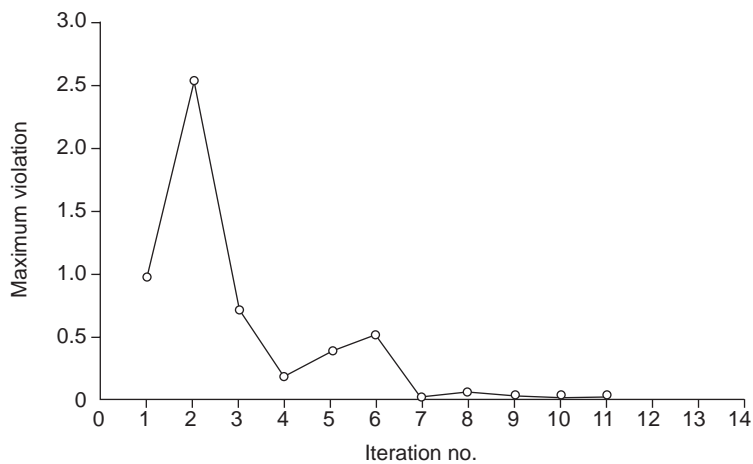


FIGURE 13-6 History of the maximum constraint violation for the spring design problem.

decreases for the first few iterations and then remains almost constant. Variable 3 does not change for the first three iterations and then increases rapidly for the next two iterations. Using the trend formation, the designer can extrapolate the value of a design variable manually.

We conclude that by using a design variable history, we can make the following decisions:

1. Based on the displayed trend, we can *extrapolate* the value of a *design variable*.
2. If a *design variable* is not changing, we can *fix* it for a few iterations and optimize only the remaining ones.

Maximum Constraint Violation History Figure 13-6 shows a plot of maximum constraint violation versus the iteration number for the spring design problem. Using this graph, we can locate feasible designs. For example, designs after iteration seven are feasible. Designs at all previous iterations had some violation of constraints.

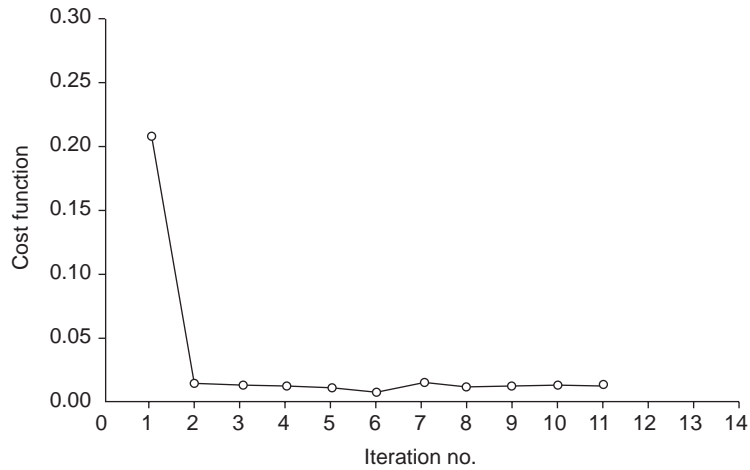


FIGURE 13-7 History of the cost function for the spring design problem.

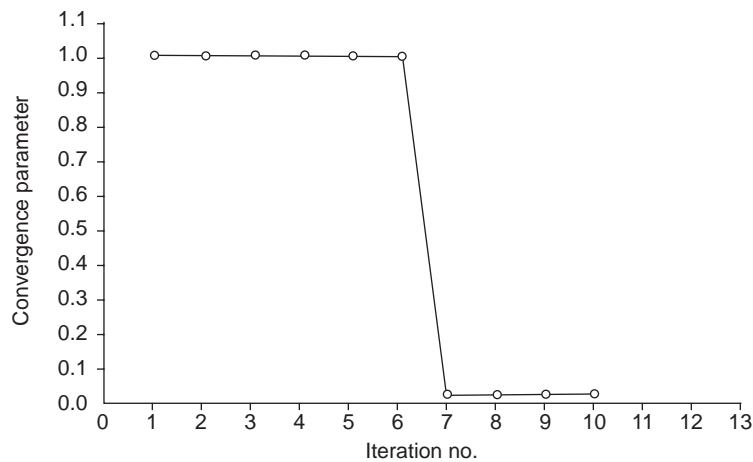


FIGURE 13-8 Convergence parameter history for the spring design problem.

Cost Function History Figure 13-7 shows the history of the cost function for the first 10 iterations for the spring design problem. The cost function decreases substantially at the first iteration. After that, it changes slowly and appears to be quite close to the optimum. The iterative process could have been terminated at a feasible design

Convergence Parameter History Figure 13-8 shows the convergence parameter history for the spring design problem. This parameter is supposed to go to zero as the optimum is reached. The parameter is close to zero at the seventh iteration, so the solution is quite close to the optimum and the iterative process could be terminated.

Constraint Function History Figure 13-9 shows the history of the four constraints for the spring design problem. A value of less than zero indicates the constraint to be inactive and greater than zero indicates the constraint to be violated. It can be seen that the first and fourth constraints are violated in the beginning, but during later iterations the first and the second

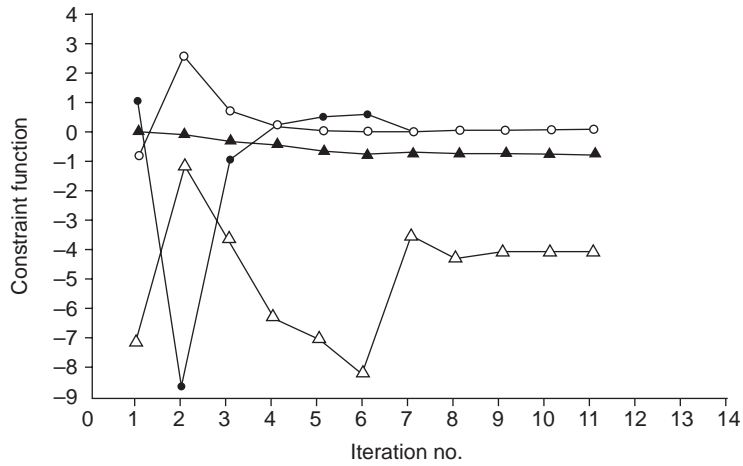


FIGURE 13-9 History of the constraint functions for the spring design problem. •, g_1 ; ○, g_2 ; △, g_3 ; ▲, g_4 .

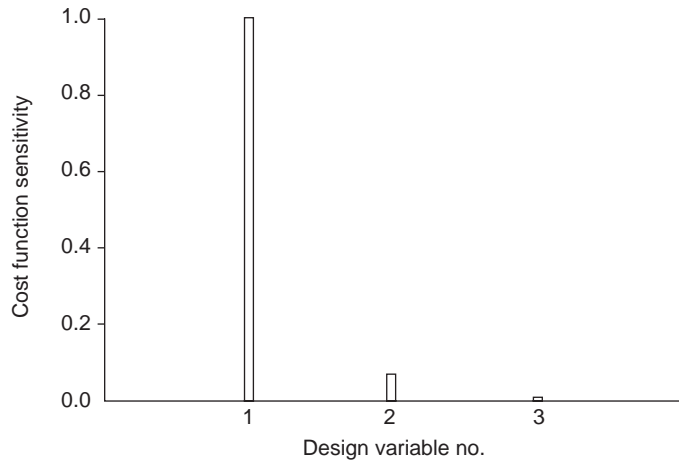


FIGURE 13-10 Cost function sensitivity to design variables for the spring design problem.

constraints are active. The third constraint is never active or violated and may be ignored. The constraint does not influence the solution or the optimization process. *Thus, using the history of constraint functions, we can identify constraints that are critical in determining the optimum solution.* The designer can further analyze these constraints to determine whether they can be adjusted to improve the optimum solution.

Cost Function Sensitivity Chart Figure 13-10 shows a normalized bar chart for the cost function sensitivity to design variables. The chart is obtained by plotting the relative values of the gradient components of the cost function (derivatives with respect to the design variables). For the spring design problem, the cost function is most sensitive to the first design variable and least sensitive to the third one. Knowing this, the designer can decide to fix the third variable and optimize only the first and the second ones.

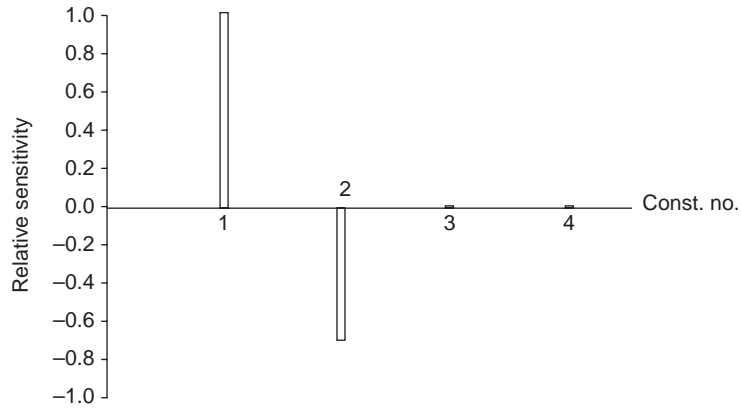


FIGURE 13-11 Sensitivity of constraints to design variable 1 for the spring design problem.

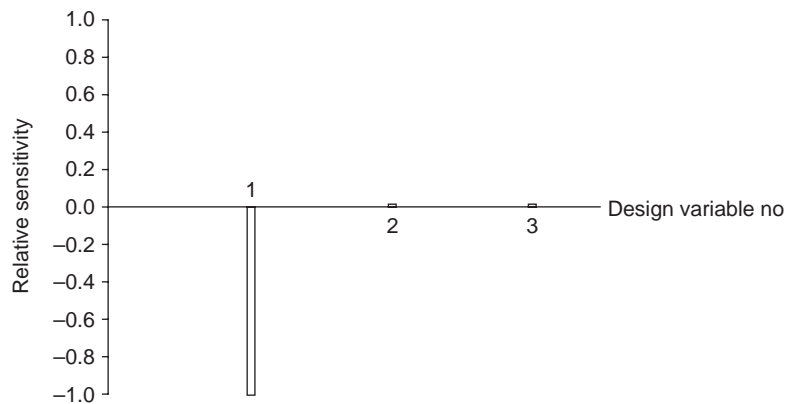


FIGURE 13-12 Sensitivity of constraint 2 to design variables for the spring design problem.

Design Variable Sensitivity for Various Constraints It may be useful to know what happens to various constraints if a design variable is changed. Figure 13-11 shows such a normalized bar chart for design variable 1 (wire diameter) for the spring design problem. It shows how the four constraints change if design variable 1 is changed slightly. For example, a small increase in the variable increases the value of the first constraint and reduces it for constraint 2. The bar chart is obtained by plotting the normalized derivatives of all the active constraints with respect to the first design variable.

Constraint Sensitivity Chart Figure 13-12 is a plot for the normalized gradient components for the second constraint for the spring design problem. It indicates what happens to the constraint function value if any design variable is changed. For example, an increase in variable 1 (wire diameter) reduces the value of the constraint quite rapidly. An increase in the values of design variable 2 increases the value of constraint 2.

Concluding Remark It can be seen that various graphs and bar charts give extremely useful information about the design problem. Such information can be used in accelerating the optimum design process as well as in learning more about the behavior of the system. The insights gained can lead to new concepts and better designs for the system.

Exercises for Chapter 13

Section 13.2 Interactive Design Optimization Algorithms

13.1 Consider the following constrained problem:

$$\text{minimize } f(\mathbf{x}) = 2x_1 + 3x_2 - x_1^3 - 2x_2^2$$

subject to

$$x_1 + 3x_2 \leq 6$$

$$5x_1 + 2x_2 \leq 10$$

$$x_1, x_2 \geq 0$$

Graph the problem functions and identify the feasible region. Locate the optimum point.

- 13.2 At the feasible point (0, 2) for Exercise 13.1, compute the cost reduction direction and obtain a new point requiring a 10 percent reduction in the cost function. Show the direction on the graph and explain your solution.
- 13.3 At the infeasible point (3, 3) for Exercise 13.1, compute the constraint correction direction, show it on the graph, and explain your solution.
- 13.4 At the infeasible point (2, 0.8) for Exercise 13.1, compute the constant cost direction, show it on the graph, and explain your solution.
- 13.5 At the infeasible point (2, 0.8) for Exercise 13.1, compute the constraint correction direction allowing only a 10 percent increase in the cost function. Show the direction on the graph and explain your solution.
- 13.6 Consider the following constrained problem:

$$\text{maximize } 2x_1 + 4x_2 - x_1^2 - x_2^2$$

subject to

$$2x_1 + 3x_2 \leq 6$$

$$x_1 + 4x_2 \leq 5$$

$$x_1, x_2 \geq 0$$

Plot the problem functions on a graph and identify the feasible region. Locate the optimum point.

- 13.7 At the feasible point (0, 1.25) for Exercise 13.6, compute the cost reduction direction and obtain a new point requiring a 10 percent reduction in the cost function. Show the direction on the graph and explain your solution.
- 13.8 At the infeasible point (4, 2) for Exercise 13.6, compute the constraint correction direction and show it on the graph. Explain your solution.

- 13.9 At the infeasible point $(1, 2)$ for Exercise 13.6, compute the constant cost direction and show it on the graph. Explain your solution.
- 13.10 At the infeasible point $(2, 2)$ for Exercise 13.6, compute the constraint correction direction allowing only a 10 percent increase in the cost function. Show the direction on the graph and explain your solution.

14 Design Optimization Applications with Implicit Functions

Upon completion of this chapter, you will be able to:

- Explain what is meant by applications that have implicit functions
- Explain how to evaluate derivatives of implicit functions for your problem
- Determine which software components need to be integrated to solve your design problem

Thus far we have considered simple engineering design problems to describe optimization concepts and computational methods. *Explicit expressions for all the functions* of the problem in terms of the design variables were assumed to be known. Whereas some practical problems can be formulated with explicit functions, there are numerous other *applications for which explicit dependence of the problem functions on design variables is not known*. In addition, complex systems require large and more sophisticated analysis models. The number of design variables and constraints can be quite large. A check for convexity of the problem is almost impossible. The existence of even feasible designs is not guaranteed, much less the optimum solution. The calculation of problem functions can require large computational effort. In many cases large special purpose software must be used to compute the problem functions. Although we will discuss some methods in Chapters 15 and 16 that do not require gradients of functions, most computational algorithms for continuous variable problems require gradients of cost and constraint functions. When an explicit form of the problem functions is not known, *gradient evaluation requires special procedures* that must be developed and implemented into proper software. Finally, various software components must also be integrated properly to create an optimum design capability for a particular class of design problems.

In this chapter, issues of the optimum design of complex practical engineering systems are addressed. Formulation of the problem, gradient evaluation, and other practical issues, such as algorithm and software selection are discussed. The important problem of interfacing a particular application with design optimization software is discussed, and several engineering design applications are described. Although most of the applications discussed in this chapter are related to mechanical and structural systems, the issues described here are

also relevant for other areas as well. Therefore, the methodologies presented and illustrated can serve as guidelines for other application areas.

14.1 Formulation of Practical Design Optimization Problems

14.1.1 General Guidelines

The problem formulation for a design task is an important step which must define a realistic model for the engineering system under consideration. The mathematics of optimization methods can easily give rise to situations that are absurd or violate the laws of physics. Therefore, to transcribe a design task correctly into a mathematical model, the designers must use intuition, skill, and experience. The following points can serve as guiding principles to generate a mathematical model that is faithful to the real-world design task.

1. Once the conceptual design is ready we must concentrate on the *detailed design of the system* to perform the given task. To start with, all the possible parameters or unknowns should be viewed as potential design variables which should be independent of each other as far as possible. Also, a variety of failure criteria and other technological requirements must be taken as constraints for the safe performance of the system. In short, considerable flexibility and freedom must be allowed before analyzing different possibilities. As we gain more knowledge about the problem, redundant or unnecessary design variables can be fixed or eliminated from the model. Finally, only significant cost and constraint functions can be retained in the design optimization model.
2. The *existence of an optimum solution* to a design optimization model depends on its formulation. If the constraints are too restrictive, there may not be any feasible solution for the problem. In such a case, the constraints must be relaxed by allowing larger resource limits for inequality constraints. The question of uniqueness of the global solution for the problem depends on the strict convexity of the cost and constraint functions. In reality, most *engineering design problems are not convex, thus global optimality of a local solution cannot be guaranteed. Usually there are multiple local optimum solutions.* This, however, is not necessarily an undesirable situation, since it offers additional freedom to the designer to choose a suitable solution among the many available ones.
3. In *numerical computations*, it is sometimes easier to find a feasible design with respect to the inequality constraints than it is with respect to the equality constraints. This, of course, depends on the problem structure and nonlinearity of functions. A constraint expressed as an inequality defines a much larger feasible region than the same constraint expressed as an equality. In case the number of equality constraints is greater than the number of design variables in a problem, no solution will exist unless some of the constraints are dependent.
4. The representation of engineering design problems by the standard nonlinear programming design optimization model with a single real valued objective function subject to a set of equality and inequality constraints is not as restrictive as it may appear. The problem of optimizing more than one objective function simultaneously (*multiobjective problems*) can be transformed into the standard problem by assigning weighting factors to different objective functions to combine them into a single objective function. Or, the most important criterion can be treated as the cost function and the remaining ones as constraints. By varying the limits for the ones treated as constraints, trade-off curves can be generated and used for the final design of the system.

5. The idea of *design variable linking* is useful to reduce the number of design variables in an optimization model. If one of the design variables can be expressed in terms of others, then that variable can be eliminated from the model. Also, if the designer can identify any symmetry in the system, it can help in reducing the number of design variables.
6. The *potential cost functions* for many structural, mechanical, automotive, and aerospace systems are weight, volume, mass, stress at a point, performance, reliability of a system, among others. The constraints can be placed on stresses, strains, displacements, vibration frequencies, manufacturing limitations, and other performance criteria.
7. It is important to have *continuous and differentiable cost and constraint functions*. If a function is not continuous or differentiable, then conventional optimization theory is not adequate. In certain instances, it may be possible to replace a nondifferentiable function such as $|x|$ with a smooth function x^2 without changing the problem definition drastically.
8. In general it is *difficult to determine dependent constraints* and eliminate them from the formulation. Modern optimization algorithms and the associated software are capable of handling difficulties arising from the dependent constraints. Also, equality constraints can be used to reduce the number of design variables by expressing one variable in terms of the others. However, such an approach is appropriate for only small-scale problems where explicit expressions for the constraints are available. In more complex applications, equality constraints must be retained and treated in the optimization algorithm.
9. In engineering design problems, *lower and upper limits on the design variables* are often imposed as a result of practical limitations. If there is no lower limit on a design variable, then a large negative number may be taken as the lower limit, and similarly a large positive number may be prescribed as the upper limit if no upper limit is given in the problem definition.
10. For nonlinear programming problems, the design variables are often assumed to be continuous. In practice, however, *discrete and integer variables* often arise. For example, because of manufacturing limitations, structural elements and spare parts for many engineering systems are available only in fixed shapes and sizes. Therefore, once we obtain the optimum solution, we can select members that have dimensions nearest to the optimum values. Or, the adaptive optimization procedure described in Section 2.11 and Chapter 15 can be used to obtain practical solutions.
11. In general, *it is desirable to normalize all the constraints with respect to their limit values*, as discussed in Section 10.1.3. In numerical computations, this procedure leads to more stable behavior. Therefore, as far as possible, all constraints should be normalized in practical applications.

14.1.2 Example of a Practical Design Optimization Problem

Optimum design formulation of complex engineering systems requires more general tools and procedures than the ones discussed previously. We shall demonstrate this by considering a class of problems that has a wide range of applications in automotive, aerospace, mechanical, and structural engineering. This important application area is chosen to *demonstrate the procedure of problem formulation and explain the treatment of implicit constraints*. Evaluation of constraint functions and their gradients shall be explained. Readers unfamiliar with this application area should use the material as *guiding principles* for their area of interest because similar analyses and procedures will need to be used in other practical applications.

The application area that we have chosen to investigate is the optimum design of systems modeled by the *finite element technique*. It is common practice to analyze complex structural systems using the technique that is also available in many commercial software packages. Displacements, stresses, and strains at various points, vibration frequencies, and buckling loads for the system can be computed and constraints imposed on them. We shall describe an optimum design formulation for this application area.

Let \mathbf{x} represent an n component vector containing *design parameters for the system*. This may contain thicknesses of members, cross-sectional areas, parameters describing the shape of the system, and stiffness and material properties of elements. Once \mathbf{x} is specified, a design of the system is known. To analyze the system (calculate stresses, strains and frequencies, buckling load, and displacements), the procedure is to first calculate displacements at some key points—called the grid points or nodal points—of the finite element model. From these displacements, strains (relative displacement of the material particles), and stresses at various points of the system are available in many textbooks (Cook, 1981; Huebner and Thornton, 1982; Grandin, 1986; Chandrupatla and Belegundu, 1997).

Let \mathbf{U} be a vector having l components representing generalized displacements at key points of the system. The basic equation that determines the displacement vector \mathbf{U} for a linear elastic system—called the equilibrium equation in terms of displacements—is given as

$$\mathbf{K}(\mathbf{x})\mathbf{U} = \mathbf{F}(\mathbf{x}) \quad (14.1)$$

where $\mathbf{K}(\mathbf{x})$ is an $l \times l$ matrix called the stiffness matrix and $\mathbf{F}(\mathbf{x})$ is an effective load vector having l components. The stiffness matrix $\mathbf{K}(\mathbf{x})$ is a *property of the structural system* that depends explicitly on the design variables, material properties, and geometry of the system. Systematic procedures have been developed to automatically calculate the matrix with different finite elements. The load vector $\mathbf{F}(\mathbf{x})$, in general, can also depend on design variables. We shall not discuss procedures to calculate $\mathbf{K}(\mathbf{x})$ because that is beyond the scope of the present text. Our objective is to demonstrate how the design can be optimized once a finite element model for the problem [meaning Eq. (14.1)] has been developed. We shall pursue that objective assuming that the finite element model for the system has been developed.

It can be seen that once the design \mathbf{x} is specified, the displacements \mathbf{U} can be calculated by solving the linear system of Eq. (14.1). Note that a different \mathbf{x} will give, in general, different values for the displacements \mathbf{U} . Thus \mathbf{U} is a function of \mathbf{x} ; however, its explicit functional form cannot be written. That is, \mathbf{U} is an *implicit function* of the design variables \mathbf{x} . The stress σ_i at the i th point is calculated using the displacements and is an explicit function of \mathbf{U} and \mathbf{x} as $\sigma_i(\mathbf{U}, \mathbf{x})$. However, since \mathbf{U} is an implicit function of \mathbf{x} , σ_i also becomes an implicit function of the design variables \mathbf{x} . The stress and displacement related constraints can be written in a functional form as

$$g_i(\mathbf{x}, \mathbf{U}) \leq 0 \quad (14.2)$$

In many automotive, aerospace, mechanical, and structural engineering applications, the amount of material used must be minimized for efficient and cost-effective systems. Thus, the usual cost function for this class of applications is the weight, mass, or material volume of the system, which is usually an explicit function of the design variables \mathbf{x} . Implicit cost functions, such as stress, displacement, vibration frequencies, etc., can also be treated by introducing artificial design variables (Haug and Arora, 1979). We shall assume that this has been done and treat only explicit cost functions.

In summary, a general formulation for the design problem involving explicit and implicit functions of design variables is defined as: Find an n -dimensional vector \mathbf{x} of design variables to minimize a cost function $f(\mathbf{x})$ satisfying the implicit design constraints of Eq. (14.2)

with \mathbf{U} satisfying the system of Eq. (14.2), and other practical limitations. Note that equality constraints, if present, can be routinely included as in the previous chapters. We illustrate the procedure of problem formulation in Example 14.1.

EXAMPLE 14.1 Design of a Two-Member Frame

Consider the design of a two-member frame subjected to out-of-plane loads as shown in Fig. 14-1. Such frames are encountered in numerous automotive, aerospace, mechanical, and structural engineering applications. We wish to formulate the problem of minimizing the volume of the frame subject to stress and size limitations (Bartel, 1969).

Solution. Since the optimum structure will be symmetric, the two members of the frame are identical. Also, it has been determined that hollow rectangular sections shall be used as members with three design variables defined as d = width of the member (in), h = height of the member (in), and t = wall thickness (in). Thus, the design variable vector is $\mathbf{x} = (d, h, t)$.

The volume of the structure is taken as the cost function, which is an explicit function of the design variables given as

$$f(\mathbf{x}) = 2L(2dt + 2ht - 4t^2) \quad (14.3)$$

To calculate stresses, we need to solve the analysis problem. The members are subjected to both bending and torsional stresses, and the combined stress constraint needs to be imposed at points 1 and 2. Let σ and τ be the maximum bending and shear stresses in the member, respectively. The failure criterion for the member is based on a combined stress theory, known as the von Mises (or maximum distortion energy) yield condition (Crandall *et al.*, 1978). With this criterion, the effective stress σ_e is given as $\sqrt{\sigma^2 + 3\tau^2}$ and the *stress constraint* is written in a normalized form as

$$\frac{1}{\sigma_a^2}(\sigma^2 + 3\tau^2) - 1.0 \leq 0 \quad (14.4)$$

where σ_a is the allowable design stress.

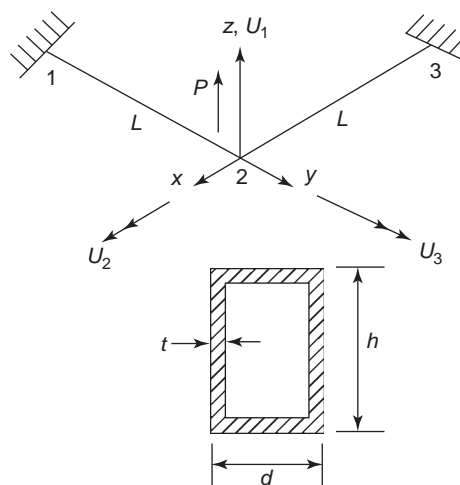


FIGURE 14-1 Two-member frame.

The stresses are calculated from the member-end moments and torques, which are calculated using the finite element procedure. The three generalized nodal displacements (deflections and rotations) for the finite element model shown in Fig. 14-1 are defined as U_1 = vertical displacement at node 2, U_2 = rotation about line 3-2, and U_3 = rotation about line 1-2. Using these, the equilibrium equation [Eq. (14.1)] for the finite element model that determines the displacements U_1 , U_2 , and U_3 , is given as (for details of the procedure to obtain the equation using individual member equilibrium equations, refer to texts by Cook, 1981; Haug and Arora, 1979; Huebner and Thornton, 1982; Grandin, 1986; Chandrupatla and Belegundu, 1997):

$$\frac{EI}{L^3} \begin{bmatrix} 24 & -6L & 6L \\ -6L \left(4L^2 + \frac{GJ}{EI} L^2 \right) & 0 & \\ 6L & 0 & \left(4L^2 + \frac{GJ}{EI} L^2 \right) \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} P \\ 0 \\ 0 \end{bmatrix} \quad (14.5)$$

where

E = modulus of elasticity, (3.0E+07) psi

L = member length, 100 in

G = shear modulus, (1.154E+07) psi

P = load at node 2, -10,000 lb

$$I = \text{moment of inertia} = \frac{1}{12} [dh^3 - (d-2t)(h-2t)^3], \text{ in}^4 \quad (14.6)$$

$$J = \text{polar moment of inertia} = \frac{2t(d-t)^2(h-t)^2}{(d+h-2t)}, \text{ in}^4 \quad (14.7)$$

A = area for calculation of torsional shear stress

$$= (d-t)(h-t), \text{ in}^2$$

From Eq. (14.5), the stiffness matrix $\mathbf{K}(\mathbf{x})$ and the load vector $\mathbf{F}(\mathbf{x})$ of Eq. (14.1) can be identified. Note that in the present example, the load vector \mathbf{F} does not depend on the design variables.

As can be seen from Eq. (14.5), \mathbf{U} is an implicit function of \mathbf{x} . If \mathbf{K} can be inverted explicitly in terms of the design variables \mathbf{x} , then \mathbf{U} can be written as an explicit function of \mathbf{x} . This is possible in the present example; however, we shall deal with the implicit form to illustrate the procedures for evaluating constraints and their gradients.

For a given design, once the displacements U_1 , U_2 , and U_3 have been calculated from Eq. (14.5), the torque and bending moment at points 1 and 2 for member 1-2 are calculated as

$$T = -\frac{GJ}{L} U_3, \quad \text{lb-in} \quad (14.8)$$

$$M_1 = \frac{2EI}{L^2} (-3U_1 + U_2L), \quad \text{lb-in (moment at end 1)} \quad (14.9)$$

$$M_2 = \frac{2EI}{L^2} (-3U_1 + 2U_2L), \quad \text{lb-in (moment at end 2)} \quad (14.10)$$

Using these moments, the torsional shear and bending stresses are calculated as

$$\tau = \frac{T}{2At}, \quad \text{psi} \quad (14.11)$$

$$\sigma_1 = \frac{1}{2I} M_1 h, \quad \text{psi (bending stress at end 1)} \quad (14.12)$$

$$\sigma_2 = \frac{1}{2I} M_2 h, \quad \text{psi (bending stress at end 2)} \quad (14.13)$$

Thus the stress constraints of Eq. (14.4) at points 1 and 2 are given as

$$g_1(\mathbf{x}, \mathbf{U}) = \frac{1}{\sigma_a^2} (\sigma_1^2 + 3\tau^2) - 1.0 \leq 0 \quad (14.14)$$

$$g_2(\mathbf{x}, \mathbf{U}) = \frac{1}{\sigma_a^2} (\sigma_2^2 + 3\tau^2) - 1.0 \leq 0 \quad (14.15)$$

We observe that since moments T , M_1 , and M_2 are implicit functions of design variables, the stresses are also implicit functions. They are also explicit functions of design variables, as seen in Eqs. (14.12) and (14.13). Therefore, the stress constraints of Eqs. (14.14) and (14.15) are implicit as well as explicit functions of design variables. This observation is important because gradient evaluation for implicit constraint functions requires special procedures, which are explained in the next section.

In addition to the two stress constraints, the following upper and lower bound constraints on design variables are imposed:

$$2.5 \leq d \leq 10.0$$

$$2.5 \leq h \leq 10.0$$

$$0.1 \leq t \leq 1.0$$

As can easily be observed, the explicit forms of the constraint functions g_1 and g_2 in terms of the design variables d , h , and t are quite difficult to obtain even for this simple problem. We will need an explicit form for the displacements U_1 , U_2 , and U_3 in Eqs. (14.8) to (14.10) to have an explicit form for the stress τ , σ_1 , and σ_2 . To have an explicit form for U_1 , U_2 , and U_3 , we will have to explicitly invert the coefficient matrix for the equilibrium equation (14.5). Although this is not impossible for the present example, it is quite impossible to do, in general. Thus we observe that the constraints are implicit functions of the design variables.

To illustrate the procedure, we select a design point as (2.5, 2.5, 0.1) and calculate the displacements and stresses. Using the given data, we calculate the following quantities that are needed in further calculations:

$$I = \frac{1}{12} [2.5^4 - 2.3^4] = 0.9232 \text{ in}^4$$

$$J = \frac{1}{4.8} [2(0.1)(2.4)^2 (2.4)^2] = 1.3824 \text{ in}^4$$

$$A = (2.4)(2.4) = 5.76 \text{ in}^2$$

$$GJ = (1.154\text{E}+07)(1.3824) = (1.5953\text{E}+07)$$

$$EI = (3.0\text{E}+07)(0.9232) = (2.7696\text{E}+07)$$

$$4L^2 + \frac{GJ}{EI} L^2 = \left(4 + \frac{1.5953}{2.7696} \right) 100^2 = (4.576\text{E}+04)$$

Using the foregoing data, the equilibrium equation (Eq. 14.5), is given as

$$27.696 \begin{bmatrix} 24 & -600 & 600 \\ -600 & 45,760 & 0 \\ 600 & 0 & 45,760 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix} = \begin{bmatrix} -10,000 \\ 0 \\ 0 \end{bmatrix}$$

Solving the preceding equation, the three generalized displacements of node 2 are given as

$$U_1 = -43.68190 \text{ in}$$

$$U_2 = -0.57275$$

$$U_3 = 0.57275$$

Using Eqs. (14.8) to (14.10), torque in the member and bending moments at points 1 and 2 are

$$T = -\frac{1.5953\text{E}+07}{100}(0.57275) = -(9.1371\text{E}+04) \text{ lb-in}$$

$$M_1 = \frac{2(2.7696\text{E}+07)}{(100)(100)} [-3(-43.68190) - 0.57275(100)] \\ = (4.0861\text{E}+05) \text{ lb-in}$$

$$M_2 = \frac{2(2.7696\text{E}+07)}{(100)(100)} [-3(-43.6819) - 2(0.57275)(100)] \\ = (9.1373\text{E}+04) \text{ lb-in}$$

Since $M_1 > M_2$, σ_1 will be larger than σ_2 as observed from Eqs. (14.12) and (14.13). Therefore, only the g_1 constraint of Eq. (14.14) needs to be imposed.

The torsional shear stress and bending stress at point 1 are calculated from Eqs. (14.11) and (14.12) as

$$\tau = \frac{-(9.13731\text{E}+04)}{2(5.76)(0.1)} = -(7.9317\text{E}+04) \text{ psi}$$

$$\sigma_1 = \frac{(4.08631\text{E}+05)(2.5)}{2(0.9232)} = (5.53281\text{E}+05) \text{ psi}$$

Taking the allowable stress σ_a as 40,000 psi, the effective stress constraint of Eq. (14.14) is given as

$$g_1 = \frac{1}{(4.0\text{E}+04)^2} [(5.53281\text{E}+05)^2 + 3(-7.9317\text{E}+04)^2] - 1 \\ = (2.0212\text{E}+02) > 0$$

Therefore, the constraint is very severely violated at the given design.

14.2 Gradient Evaluation for Implicit Functions

To use the modern optimization method, we need to evaluate gradients of constraint functions. When the constraint functions are implicit in design variables, we need to develop and utilize special procedures for gradient evaluation. We shall develop a procedure using the finite element application of Section 14.1.

Let us consider the constraint function $g_i(\mathbf{x}, \mathbf{U})$ of Eq. (14.2). Using the chain rule of differentiation, the total derivative of g_i with respect to the j th design variable is given as

$$\frac{dg_i}{dx_j} = \frac{\partial g_i}{\partial x_j} + \frac{\partial g_i^T}{\partial \mathbf{U}} \frac{d\mathbf{U}}{dx_j} \quad (14.16)$$

where

$$\frac{\partial g_i}{\partial \mathbf{U}} = \left[\frac{\partial g_i}{\partial U_1} \quad \frac{\partial g_i}{\partial U_2} \quad \cdots \quad \frac{\partial g_i}{\partial U_l} \right]^T \quad (14.17)$$

and

$$\frac{d\mathbf{U}}{dx_j} = \left[\frac{\partial U_1}{\partial x_j} \quad \frac{\partial U_2}{\partial x_j} \quad \cdots \quad \frac{\partial U_l}{\partial x_j} \right]^T \quad (14.18)$$

Therefore, to calculate the gradient of a constraint, we need to calculate the partial derivatives $\partial g_i/\partial x_j$ and $\partial g_i/\partial \mathbf{U}$, and the total derivatives $d\mathbf{U}/dx_j$. The partial derivatives $\partial g_i/\partial x_j$ and $\partial g_i/\partial \mathbf{U}$ are quite easy to calculate using the form of the function $g_i(\mathbf{x}, \mathbf{U})$. To calculate $d\mathbf{U}/dx_j$, we differentiate the equilibrium Eq. (14.1) to obtain

$$\frac{\partial \mathbf{K}(\mathbf{x})}{\partial x_j} \mathbf{U} + \mathbf{K}(\mathbf{x}) \frac{d\mathbf{U}}{dx_j} = \frac{\partial \mathbf{F}}{\partial x_j} \quad (14.19)$$

Or, the equation can be rearranged as

$$\mathbf{K}(\mathbf{x}) \frac{d\mathbf{U}}{dx_j} = \frac{\partial \mathbf{F}}{\partial x_j} - \frac{\partial \mathbf{K}(\mathbf{x})}{\partial x_j} \mathbf{U} \quad (14.20)$$

The equation can be used to calculate $d\mathbf{U}/dx_j$. The derivative of the stiffness matrix $\partial \mathbf{K}(\mathbf{x})/\partial x_j$ can easily be calculated if the explicit dependence of \mathbf{K} on \mathbf{x} is known. Note that Eq. (14.20) needs to be solved for each design variable. Once $d\mathbf{U}/dx_j$ are known, the gradient of the constraint is calculated from Eq. (14.16). The derivative vector in Eq. (14.16) is often called the design gradient. We shall illustrate the procedure with an example problem.

It should be noted that substantial work has been done in developing and implementing efficient procedures for calculating derivatives of implicit functions with respect to the design variables (Arora and Haug, 1979; Adelman and Haftka, 1986; Arora, 1995). The subject is generally known as *design sensitivity analysis*. For efficiency considerations and proper numerical implementations, the foregoing literature should be consulted. The procedures have been programmed into general-purpose software for automatic computation of design gradients.

EXAMPLE 14.2 Gradient Evaluation for a Two-Member Frame

Calculate the gradient of the stress constraint $g_1(\mathbf{x}, \mathbf{U})$ for the two-member frame of Example 14.1 at the design point (2.5, 2.5, 0.1).

Solution. The problem has been formulated in Example 14.1. The finite element model has been defined there, and nodal displacements and member stresses have been calculated. We shall use Eqs. (14.16) and (14.20) to evaluate the gradient of the stress constraint of Eq. (14.14).

The partial derivatives of the constraint of Eq. (14.14) with respect to \mathbf{x} and \mathbf{U} are given as

$$\frac{\partial g_1}{\partial \mathbf{x}} = \frac{1}{\sigma_a^2} \left[2\sigma_1 \frac{\partial \sigma_1}{\partial \mathbf{x}} + 6\tau \frac{\partial \tau}{\partial \mathbf{x}} \right] \quad (14.21)$$

$$\frac{\partial g_1}{\partial \mathbf{U}} = \frac{1}{\sigma_a^2} \left[2\sigma_1 \frac{\partial \sigma_1}{\partial \mathbf{U}} + 6\tau \frac{\partial \tau}{\partial \mathbf{U}} \right] \quad (14.22)$$

Using Eqs. (14.8) to (14.13), partial derivatives of τ and σ_1 with respect to \mathbf{x} and \mathbf{U} are calculated as follows.

Partial derivatives of shear stress. Differentiating the expression for shear stress in Eq. (14.11) with respect to the design variables \mathbf{x} , we get

$$\frac{\partial \tau}{\partial \mathbf{x}} = \frac{1}{2At} \frac{\partial T}{\partial \mathbf{x}} - \frac{T}{2A^2t} \frac{\partial A}{\partial \mathbf{x}} - \frac{T}{2At^2} \frac{\partial t}{\partial \mathbf{x}} \quad (14.23)$$

where partial derivatives of the torque T with respect to the design variable \mathbf{x} are given as

$$\frac{\partial T}{\partial \mathbf{x}} = -\frac{GU_3}{L} \frac{\partial J}{\partial \mathbf{x}} \quad (14.24)$$

with $\partial J/\partial \mathbf{x}$ calculated as

$$\frac{\partial J}{\partial d} = \frac{4t(d-t)(h-t)^2(d+h-2t) - 2t(d-t)^2(h-t)^2}{(d+h-2t)^2} = 0.864$$

$$\frac{\partial J}{\partial h} = \frac{4t(d-t)^2(h-t)(d+h-2t) - 2t(d-t)^2(h-t)^2}{(d+h-2t)^2} = 0.864$$

$$\frac{\partial J}{\partial t} = \frac{2(d-t)^2(h-t)^2 - 4t(d-t)(h-t)^2 - 4t(d-t)^2(h-t)}{(d+h-2t)}$$

$$= \frac{2t(d-t)^2(h-t)^2(-2)}{(d+h-2t)^2} = 12.096$$

Therefore, $\partial J/\partial \mathbf{x}$ is assembled as

$$\frac{\partial J}{\partial \mathbf{x}} = \begin{bmatrix} 0.864 \\ 0.864 \\ 12.096 \end{bmatrix}$$

and Eq. (14.24) gives $\partial T/\partial \mathbf{x}$ as

$$\begin{aligned} \frac{\partial T}{\partial \mathbf{x}} &= -\frac{(1.154\text{E}+07)}{100}(0.57275) \begin{bmatrix} 0.864 \\ 0.864 \\ 12.096 \end{bmatrix} \\ &= -(6.610\text{E}+04) \begin{bmatrix} 0.864 \\ 0.864 \\ 12.096 \end{bmatrix} \end{aligned}$$

Other quantities needed to complete calculations in Eq. (14.23) are $\partial A/\partial \mathbf{x}$ and $\partial t/\partial \mathbf{x}$, which are calculated as

$$\begin{aligned} \frac{\partial A}{\partial \mathbf{x}} &= \begin{bmatrix} (h-t) \\ (d-t) \\ -(h-t)-(d-t) \end{bmatrix} = \begin{bmatrix} 2.4 \\ 2.4 \\ -4.6 \end{bmatrix} \\ \frac{\partial t}{\partial \mathbf{x}} &= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \end{aligned}$$

Substituting various quantities into Eq. (14.23), we get the partial derivative of τ with respect to \mathbf{x} as

$$\frac{\partial \tau}{\partial \mathbf{x}} = \frac{1}{2At} \left[\frac{\partial T}{\partial \mathbf{x}} - \frac{T}{A} \frac{\partial A}{\partial \mathbf{x}} - \frac{T}{t} \frac{\partial t}{\partial \mathbf{x}} \right] = \begin{bmatrix} -1.653\text{E}+04 \\ -1.653\text{E}+04 \\ 3.580\text{E}+04 \end{bmatrix}$$

Differentiating the expression for the shear stress τ in Eq. (14.11) with respect to the generalized displacements \mathbf{U} , we get

$$\frac{\partial \tau}{\partial \mathbf{U}} = \frac{1}{2At} \frac{\partial T}{\partial \mathbf{U}}$$

where

$$\frac{\partial T}{\partial \mathbf{U}} = \begin{bmatrix} 0 \\ 0 \\ -GJ/L \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1.5953\text{E}+05 \end{bmatrix}$$

Therefore, $\partial \tau/\partial \mathbf{U}$ is given as

$$\frac{\partial \tau}{\partial \mathbf{U}} = \begin{bmatrix} 0 \\ 0 \\ -1.3848\text{E}+05 \end{bmatrix}$$

Partial derivatives of bending stress. Differentiating the expression for σ_1 given in Eq. (14.12) with respect to design variables \mathbf{x} , we get

$$\frac{\partial \sigma_1}{\partial \mathbf{x}} = \frac{h}{2I} \frac{\partial M_1}{\partial \mathbf{x}} + \frac{M_1}{2I} \frac{\partial h}{\partial \mathbf{x}} - \frac{M_1 h}{2I^2} \frac{\partial I}{\partial \mathbf{x}} \quad (14.25)$$

where $\partial M_1/\partial \mathbf{x}$, $\partial I/\partial \mathbf{x}$, and $\partial h/\partial \mathbf{x}$ are given as

$$\begin{aligned} \frac{\partial M_1}{\partial \mathbf{x}} &= \frac{2E}{L^2} (-3U_1 + U_2 L) \frac{\partial I}{\partial \mathbf{x}} \\ \frac{\partial I}{\partial d} &= \frac{1}{12} [h^3 - (h-2t)^3] \\ &= 0.288167 \\ \frac{\partial I}{\partial h} &= \frac{1}{4} [dh^2 - (d-2t)(h-2t)^2] \\ &= 0.8645 \\ \frac{\partial I}{\partial t} &= \frac{(h-2t)^3}{6} + \frac{(h-2t)^2(d-2t)}{2} \\ &= 8.11133 \\ \frac{\partial h}{\partial \mathbf{x}} &= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \end{aligned}$$

Substituting various quantities into Eq. (14.25),

$$\frac{\partial \sigma_1}{\partial \mathbf{x}} = \begin{bmatrix} 0 \\ 2.2131\text{E}+05 \\ 0 \end{bmatrix}$$

Differentiating the expression for σ_1 in Eq. (14.12) with respect to generalized displacements \mathbf{U} , we get

$$\frac{\partial \sigma_1}{\partial \mathbf{U}} = \frac{h}{2I} \frac{\partial M_1}{\partial \mathbf{U}}$$

where $\partial M_1/\partial \mathbf{U}$ is given from Eq. (14.9) as

$$\frac{\partial M_1}{\partial \mathbf{U}} = \frac{2EI}{L^2} \begin{bmatrix} -3 \\ L \\ 0 \end{bmatrix}$$

Therefore, $\partial \sigma_1/\partial \mathbf{U}$ is given as

$$\frac{\partial \sigma_1}{\partial \mathbf{U}} = \frac{Eh}{L^2} \begin{bmatrix} -3 \\ L \\ 0 \end{bmatrix} = \begin{bmatrix} -2.25\text{E}+05 \\ 7.50\text{E}+05 \\ 0 \end{bmatrix}$$

Substituting various quantities into Eqs. (14.21) and (14.22), we obtain the partial derivatives of constraints as

$$\frac{\partial g_1}{\partial \mathbf{x}} = \begin{bmatrix} 4.917 \\ 157.973 \\ -10.648 \end{bmatrix}$$

$$\frac{\partial g_1}{\partial \mathbf{U}} = \begin{bmatrix} -15.561 \\ 518.700 \\ 41.190 \end{bmatrix}$$

Derivatives of the displacements. To calculate derivatives of the displacements, we use Eq. (14.20). Since the load vector does not depend on design variables, $\partial \mathbf{F}/\partial x_j = 0$ for $j = 1, 2, 3$ in Eq. (14.20). To calculate $(\partial \mathbf{K}(\mathbf{x})/\partial x_j)\mathbf{U}$ on the right-side of Eq. (14.20), we differentiate Eq. (14.5) with respect to the design variables. For example, differentiation of Eq. (14.5) with respect to d gives the following vector:

$$\frac{\partial \mathbf{K}(\mathbf{x})}{\partial d} \mathbf{U} = \begin{bmatrix} -3.1214\text{E}+03 \\ -2.8585\text{E}+04 \\ 2.8585\text{E}+04 \end{bmatrix}$$

Similarly, by differentiating with respect to h and t , we obtain

$$\frac{\partial \mathbf{K}(\mathbf{x})}{\partial \mathbf{x}} \mathbf{U} = (1.0\text{E}+03) \begin{bmatrix} -3.1214 & -9.3642 & -87.861 \\ -28.5850 & 28.4540 & 3.300 \\ 28.5850 & -28.4540 & -3.300 \end{bmatrix}$$

Since $\mathbf{K}(\mathbf{x})$ is already known in Example 14.1, we use Eq. (14.20) to calculate $d\mathbf{U}/d\mathbf{x}$ as

$$\frac{d\mathbf{U}}{d\mathbf{x}} = \begin{bmatrix} 16.9090 & 37.6450 & 383.4200 \\ 0.2443 & 0.4711 & 5.0247 \\ -0.2443 & -0.4711 & -5.0247 \end{bmatrix}$$

Finally, substituting all the quantities in Eq. (14.16), we obtain the design gradient for the effective stress constrain of Eq. (14.14) as

$$\frac{dg_1}{d\mathbf{x}} = \begin{bmatrix} -141.55 \\ -202.87 \\ -3577.30 \end{bmatrix}$$

As noted in Example 14.1, the stress constraint of Eq. (14.14) is severely violated at the given design. The signs of the foregoing design derivatives indicate that all the variables will have to be increased to reduce the constraint violation at the point (2.5, 2.5, 0.1).

14.3 Issues in Practical Design Optimization

Several issues need to be considered for practical design optimization. For example, careful consideration needs to be given to the selection of an algorithm and the software. Improper choice of either one can mean failure of the optimum design process and frustration with optimization techniques. In this section, we shall discuss some of the issues that can have a significant impact on practical applications of the optimization methodology.

14.3.1 Selection of an Algorithm

Many algorithms have been developed and evaluated for practical design optimization. We need to consider several aspects when selecting an algorithm for practical applications, such as robustness, efficiency, generality, and ease of use.

Robustness Characteristics of a robust algorithm are discussed in Section 10.1.5. For practical applications, it is extremely important to use a method that is theoretically guaranteed to converge. *A method having such a guarantee starting from any initial design estimate is called robust.* Robust algorithms usually require a few more calculations during each iteration compared with algorithms that have no proof of convergence. Such approximate algorithms usually require considerable tuning for each problem before a reasonable optimum solution is obtained, and many times they converge to nonoptimum solutions. Thus, although the approximate algorithms consume slightly less computer time during each iteration, they have considerable uncertainty in their performance. They usually need considerable experimentation and tuning to make them work for a particular problem. This can be an unnecessary distraction requiring the designer's time. Therefore, in the overall sense, such approximate algorithms are actually more costly to use than the robust algorithms. We suggest that only robust algorithms be used in practical applications, i.e., robustness must be given higher priority over efficiency while selecting an algorithm and the associated software.

Potential Constraint Strategy To evaluate the search direction in numerical methods for constrained optimization, we need to know the cost and constraint functions and their gradients. *The numerical algorithms can be classified into two categories based on whether gradients of all the constraints or only a subset of them are required during a design iteration.* The numerical algorithms that need the gradients of only a subset of the constraints are said to use *potential constraint strategy*. The potential constraint set, in general, is composed of active, nearly active, and violated constraints at the current iteration. For a further discussion on the topic of potential set strategy, refer to Section 11.1.

14.3.2 Attributes of a Good Optimization Algorithm

Based on the preceding discussion, attributes of a good algorithm for practical design applications are defined as follows.

1. *Reliability.* The algorithm must be reliable for general design applications because such algorithms converge to a minimum point starting from any initial design estimate. Reliability of an algorithm is guaranteed if it is theoretically proven to converge.
2. *Generality.* The algorithm must be general, which implies that it should be able to treat equality as well as inequality constraints. In addition, it should not impose any restrictions on the form of functions of the problem.
3. *Ease of use.* The algorithm must be easy to use by the experienced as well as the inexperienced designer. From a practical standpoint, this is an extremely important requirement because an algorithm requiring selection of tuning parameters is quite

difficult to use. The proper specification of the parameters usually requires not only a complete knowledge and understanding of the mathematical structure of the algorithm but also experimentation with each problem. Such an algorithm is unsuitable for practical design applications.

4. *Efficiency.* The algorithm must be efficient for general engineering applications. An efficient algorithm has (1) a faster rate of convergence to the minimum point, and (2) the fewest number of calculations within one design iteration. The *rate of convergence* can be accelerated by incorporating the second-order information about the problem into the algorithm. Incorporation of the second-order information, however, requires additional calculations during an iteration. Therefore, there is a trade-off between efficiency of calculation within an iteration and the rate of convergence. Some existing algorithms use second-order information whereas others do not. Efficiency within an iteration implies a minimum number of calculations for the search direction and the step size. One way to achieve efficiency is to use a *potential constraint strategy* in calculating the search direction. There are some algorithms that use this strategy in their calculations while others do not. When the potential constraint strategy is used, the direction-finding subproblem needs gradients of only potential constraints. Otherwise, gradients of all constraints are needed, which is inefficient in most practical applications.

Another consideration for improving efficiency within an iteration is to keep the number of function evaluations for step size determination to a minimum. This can be achieved by using step size determination procedures requiring fewer calls for function evaluations, e.g., inaccurate line search.

The designer needs to ask the following questions (all answers should be yes) before selecting an optimization algorithm for practical applications:

1. Does the algorithm have proof of convergence? That is, is it theoretically guaranteed to converge to an optimum point starting from any initial design estimate? Can the starting design be infeasible (i.e., arbitrary)?
2. Can the algorithm solve a general optimization problem without any restrictions on the constraint functions? Can it treat equality as well as inequality constraints?
3. Is the algorithm easy to use? In other words, it does not require tuning for each problem?
4. Does the algorithm incorporate a potential constraint strategy?

14.4 Use of General-Purpose Software

As we have seen in previous sections, practical systems require considerable computer analysis before optimum solutions are obtained. For a particular application, problem functions and gradient evaluation software as well as optimization software must be integrated to create an optimum design capability. Depending on the application, each of the software components can be very large. Therefore, to create a design optimization capability, the most sophisticated and modern computer facilities must be used to integrate the software components.

For the example of structures modeled by the finite elements discussed in Section 14.1.2, large analysis packages must be used to analyze the structure. From the calculated response, constraint functions must be evaluated and programs must be developed to calculate gradients. All the software components must be integrated to create the optimum design capability for structures modeled by finite elements.

In this section we shall discuss the issues involved in selecting a general-purpose optimization software. Also interfacing of the software with a particular application shall be discussed.

14.4.1 Software Selection

Several issues need to be explored before general-purpose optimization software is selected for integration with other application-dependent software. The most important question pertains to the optimization algorithm and how well it is implemented. The attributes of a good algorithm are given in Section 14.3.2. The software must contain at least one algorithm that satisfies all the requirements stated there. The algorithm should also be robustly implemented because a good algorithm when badly implemented is not very useful. The proof of convergence of most algorithms is based on certain assumptions. These need to be strictly adhered to while implementing the algorithm. In addition, most algorithms have numerical uncertainties in their steps which need to be recognized and proper procedures need to be developed for numerical implementation. It is also important that the software be well tested on a range of applications of varying difficulty.

Several other user-friendly features are desirable. For example, the possibility of interaction during the iterative process, interactive graphics, and other aids to facilitate design decision making are highly desirable. The topic of interactive design optimization and facilities is covered in more detail in Chapter 13. Documentation for the software is also very important. How good is the user's manual? What sample problems are available with the program and how well are they documented? How easy is it to install the program on different computer systems? All these questions should be investigated before selecting the software.

14.4.2 Integration of an Application into General-Purpose Software

Each general-purpose program for optimization requires that the particular design application be integrated into the software. Ease of integration of the software components for various applications can influence selection of the program. Also, the amount of data preparation needed to use the program is important.

Some general-purpose libraries are available that contain various subroutines implementing different algorithms. The user is required to write a main program to call the subroutine. Subprograms for function and gradient evaluation must also be written. Several exits are made from the subroutine, and it is re-entered with different conditions. The main program becomes quite complex and is prone to user error. In addition, there is little chance for user interaction in this environment because users must develop their own interactive capability.

The other approach is to develop a computer program with options of various algorithms and facilities. Each application is implanted into the program through a standard interface that consists of "subroutine calls." The user prepares a few subroutines to describe the design problem only. All the data between the program and the subroutines flow through the subroutine arguments. For example, design variable data are sent to the subroutines and the expected output is the constraint function values and their gradients. Most interactive capabilities, graphics, and other user-friendly features are built into the program. A key feature of this approach is that the users do not get involved with the algorithmic idiosyncrasies in selecting various parameters and trying to make the algorithm work. They are relieved of these duties by the software developers, and can concentrate on their design problem formulation and its description for the program.

Both procedures described in the foregoing paragraphs have been successfully used in the past for many practical applications of optimization. In most cases, the choice of the procedure has been dictated by the availability of the software. We shall use the program IDESIGN, which is based on the second approach, to solve several design optimization problems. Figure

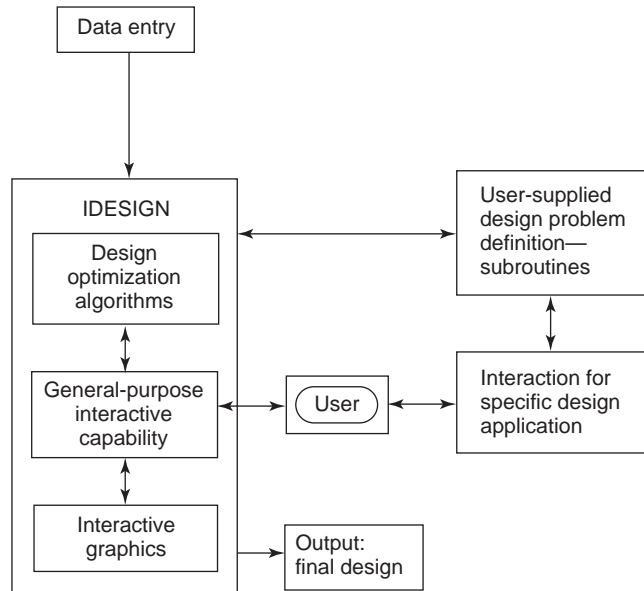


FIGURE 14-2 Implantation of a design application into IDESIGN. IDESIGN is combined with user-supplied subroutines to create an executable module.

14-2 shows how an application can be implanted into the program IDESIGN. As explained in Section 13.4, the program is combined with the user-supplied subroutines to create an executable module. *The user-supplied subroutines can be quite simple for problems having explicit functions and complex for problems having implicit functions. External subroutines or programs may have to be called upon to generate the function values and their gradients needed by IDESIGN.* This has been done and several complex problems have been solved and reported in the literature (Tseng and Arora, 1987, 1988).

Besides the general-purpose interactive capability contained in IDESIGN, the user can write additional interactive facilities for the specific application, as shown in Fig. 14-2. Input data for the application are supplied to the program to execute it. Using this arrangement several practical problems from different fields have been solved with the program. In the remaining sections of this chapter, we shall describe several design problems and solve them with the previously mentioned arrangement for the program IDESIGN.

14.5 Optimum Design of a Two-Member Frame with Out-of-Plane Loads

Figure 14-1 shows a two-member frame subjected to out-of-plane loads. The members of the frame are subjected to torsional, bending, and shearing loads. The objective of the problem is to design a structure having minimum volume without material failure due to applied loads. The problem has been formulated in Section 14.1.2 using the finite element approach. In defining the stress constraint, von Mises yield criterion is used and the shear stress due to the transverse load is neglected.

The formulation and equations give in Sections 14.1.2 and 14.2 are used to develop appropriate subroutines for the program IDESIGN. The data given there are used to optimize the problem. Two widely separated starting designs, (2.5, 2.5, 0.1) and (10, 10, 1), are tried to observe their effect on the convergence rate. For the first starting point, all the variables are

at their lower bounds; for the second point they are all at their upper bounds. Both starting points converge to the same optimum solution with almost the same values for design variables and Lagrange multipliers for active constraints, as shown in Tables 14-1 and 14-2 (refer to Section 13.5 for an explanation of the notations used in the tables). However, the number of iterations and the number of calls for function and gradient evaluations are quite different. For the first starting point, the stress constraint is severely violated (by 20,212 percent). Several iterations are expended to bring the design close to the feasible region. For the second starting point, the stress constraint is satisfied and the program takes only six iterations to find the optimum solution. Note that both solutions reported in Tables 14-1 and 14-2 are obtained by using the *Sequential Quadratic Programming* option (SQP) available in IDESIGN. Also, a very severe convergence criterion is used to obtain the precise optimum point.

The preceding discussion shows that the starting design estimate for the iterative process can have a substantial impact on the rate of convergence of an algorithm. In many practical applications, a good starting design is available or can be obtained after some preliminary analyses. Such a starting design for the optimization algorithm is desirable because this can save substantial computational effort.

TABLE 14-1 History of the Iterative Process and Optimum Solution for a Two-Member Frame, Starting Point (2.5, 2.5, 0.10)

<i>I</i>	<i>Max. vio.</i>	<i>Conv. parm.</i>	<i>Cost</i>	<i>d</i>	<i>h</i>	<i>t</i>
1	2.02119E+02	1.00000E+00	1.92000E+02	2.5000E+00	2.5000E+00	1.0000E-01
2	8.57897E+01	1.00000E+00	2.31857E+02	2.5000E+00	3.4964E+00	1.0000E-01
3	3.58717E+01	1.00000E+00	2.85419E+02	2.5000E+00	4.8355E+00	1.0000E-01
⋮	⋮	⋮	⋮	⋮	⋮	⋮
17	6.78824E-01	1.00000E+00	6.14456E+02	5.5614E+00	1.0000E+01	1.0000E-01
18	1.58921E-01	6.22270E-01	6.76220E+02	7.1055E+00	1.0000E+01	1.0000E-01
19	1.47260E-02	7.01249E-02	7.01111E+02	7.7278E+00	1.0000E+01	1.0000E-01
20	1.56097E-04	7.59355E-04	7.03916E+02	7.7979E+00	1.0000E+01	1.0000E-01

Constraint activity

<i>No.</i>	<i>Active</i>	<i>Value</i>	<i>Lagr. mult.</i>
1	Yes	1.56097E-04	1.94631E+02

Design variable activity

<i>No.</i>	<i>Active</i>	<i>Design</i>	<i>Lower</i>	<i>Upper</i>	<i>Lagr. mult.</i>
1	No	7.79791E+00	2.50000E+00	1.00000E+01	0.00000E+00
2	Upper	1.00000E+01	2.50000E+00	1.00000E+01	7.89773E+01
3	Lower	1.00000E-01	1.00000E-01	1.00000E+00	3.19090E+02

Cost function at optimum = 7.039163E+02

No. of calls for cost function evaluation = 20

No. of calls for evaluation of cost function gradient = 20

No. of calls for constraint function evaluation = 20

No. of calls for evaluation of constraint function gradients = 20

No. of total gradient evaluations = 20

TABLE 14-2 History of the Iterative Process and Optimum Solution for a Two-Member Frame, Starting Point (10, 10, 1)

<i>I</i>	<i>Max. vio.</i>	<i>Conv. parm.</i>	<i>Cost</i>	<i>d</i>	<i>h</i>	<i>t</i>
1	0.00000E+00	6.40000E+03	7.20000E+03	1.0000E+01	1.0000E+01	1.0000E+00
2	0.00000E+00	2.27873E+01	7.87500E+02	9.9438E+00	9.9438E+00	1.0000E-01
3	1.25020E-02	1.31993E+00	7.13063E+02	9.0133E+00	9.0133E+00	1.0000E-01
4	2.19948E-02	1.03643E-01	6.99734E+02	7.6933E+00	1.0000E+01	1.0000E-01
5	3.44115E-04	1.67349E-03	7.03880E+02	7.7970E+00	1.0000E+01	1.0000E-01
6	9.40469E-08	4.30513E-07	7.03947E+02	7.7987E+00	1.0000E+01	1.0000E-01

Constraint activity

<i>No.</i>	<i>Active</i>	<i>Value</i>	<i>Lagr. mult.</i>
1	Yes	9.40469E-08	1.94630E+02

Design variable activity

<i>No.</i>	<i>Active</i>	<i>Design</i>	<i>Lower</i>	<i>Upper</i>	<i>Lagr. mult.</i>
1	No.	7.79867E+00	2.50000E+00	1.00000E+01	0.00000E+00
2	Upper	1.00000E+01	2.50000E+00	1.00000E+01	7.89767E+01
3	Lower	1.00000E-01	1.00000E-00	1.00000E+00	3.19090E+02

Cost function at optimum = 7.039466E+02
 No. of calls for cost function evaluation = 9
 No. of calls for evaluation of cost function gradient = 6
 No. of calls for constraint function evaluation = 9
 No. of calls for evaluation of constraint function gradient = 4
 No. of total gradient evaluations = 4

14.6 Optimum Design of a Three-Bar Structure for Multiple Performance Requirements

In the previous section, we discussed design of a structural system for one performance requirement—the material must not fail under the applied loads. In this section, we discuss a similar application where the system must perform safely under several operating environments. The problem that we have chosen is the three-bar structure that is formulated in Section 2.10. The structure is shown in Fig. 2-6. The design requirement is to minimize the weight of the structure and satisfy the constraints of member stress, deflection at node 4, buckling of members, vibration frequency, and explicit bounds on the design variables. We shall optimize symmetric and asymmetric structures and compare the solutions. A very strict convergence criterion shall be used to obtain the precise optimum designs.

14.6.1 Symmetric Three-Bar Structure

A detailed formulation for the symmetric structure where members 1 and 3 are similar is discussed in Section 2.10. In the present application, the structure is designed to withstand three loading conditions and the foregoing constraints. Table 14-3 contains all the data used for designing the structure. All the expressions programmed for IDESIGN are given in Section 2.10. The constraint functions are appropriately normalized and expressed in the standard form.

TABLE 14-3 Design Data for a Three-Bar Structure

Allowable stress:	Members 1 and 3, $\sigma_{1a} = \sigma_{3a} = 5000$ psi Member 2, $\sigma_{2a} = 20,000$ psi		
Allowable displacements:	$u_a = 0.005$ in $v_a = 0.005$ in		
Modulus of elasticity:	$E = (1.00E+07)$ psi		
Weight density:	$\gamma = (1.00E-01)$ lb/in ³		
Constant:	$\beta = 1.0$		
Lower limit on design:	$(0.1, 0.1, 0.1)$ in ²		
Upper limit on design:	$(100, 100, 100)$ in ²		
Starting design:	$(1, 1, 1)$ in ²		
Lower limit on frequency:	2500 Hz		
Loading conditions:	3		
Angle, θ (degrees)	45	90	135
Load, P (lb)	40,000	30,000	20,000

To study the effect of imposing more performance requirements, the following three design cases are defined (note that explicit design variable bound constraints are also imposed in all cases):

Case 1. Stress constraints only (total constraints = 13).

Case 2. Stress and displacement constraints (total constraints = 19).

Case 3. All constraints—stress, displacement, member buckling, and frequency (total constraints = 29).

Tables 14-4 to 14-6 contain the history of the iterative process, constraint and design variable activities at the final design, and the optimum cost function for the three cases with the SQP method in IDESIGN. The active constraints at the optimum point and their Lagrange multipliers (for normalized constraints) are:

Case 1. Stress in member 1 under loading condition 1, 21.11.

Case 2. Stress in member 1 under loading condition 1, 0.0; horizontal displacement under loading condition 1, 16.97; vertical displacement under loading condition 1, 6.00; horizontal displacement under loading condition 2, 0.0.

Case 3. Same as for Case 2.

Note that the cost function value at the optimum point increases for Case 2 as compared with Case 1. This is consistent with the hypothesis that more constraints for the system imply a smaller feasible region, thus giving a higher value for the optimum cost function. There is no difference between the solutions for Cases 2 and 3 because none of the additional constraints for Case 3 is active.

14.6.2 Asymmetric Three-Bar Structure

When the symmetry condition for the structure (member 1 same as member 3) is relaxed, we get three design variables for the problem compared with only two for the symmetric

TABLE 14-4 History of the Iterative Process and Final Solution for a Symmetric Three-Bar Structure, Case 1—Stress Constraints

<i>I</i>	<i>Max. vio.</i>	<i>Conv. parm.</i>	<i>Cost</i>	$A_1 = A_3$	A_2
1	4.65680E+00	1.00000E+00	3.82843E+00	1.0000E+00	1.0000E+00
2	2.14531E+00	1.00000E+00	6.72082E+00	1.9528E+00	1.1973E+00
⋮	⋮	⋮	⋮	⋮	⋮
8	2.20483E-04	3.97259E-03	2.11068E+01	6.3140E+00	3.2482E+00
9	1.58618E-06	5.34172E-05	2.11114E+01	6.3094E+00	3.2657E+00

Constraint activity

No.	Active	Value	Lagr. mult.
1	Yes	1.58618E-06	2.11114E+01
2	No	-7.32069E-01	0.00000E+00
3	No	-8.16982E-01	0.00000E+00
4	No	-6.11757E-01	0.00000E+00
5	No	-6.11757E-01	0.00000E+00
6	No	-8.05879E-01	0.00000E+00
7	No	-8.66035E-01	0.00000E+00
8	No	-4.99999E-01	0.00000E+00
9	No	-9.08491E-01	0.00000E+00

Design variable activity

No.	Active	Design	Lower	Upper	Lagr. mult.
1	No	6.30942E+00	1.00000E-01	1.00000E+02	0.00000E+00
2	No	3.26569E+00	1.00000E-01	1.00000E+02	0.00000E+00

Cost function at optimum = 2.111143E+01
 No. of calls for cost function evaluation = 9
 No. of calls for evaluation of cost function gradient = 9
 No. of calls for constraint function evaluation = 9
 No. of calls for evaluation of constraint function gradient = 9
 No. of total gradient evaluations = 19

case, i.e., areas A_1 , A_2 , and A_3 for members 1, 2, and 3, respectively. With this, the design space becomes expanded so we can expect better optimum designs compared with the previous cases. The data used for the problem are the same as given in Table 14-3. The structure is optimized for the following three cases (note that the explicit design variable bound constraints are imposed in all cases):

- Case 4.* Stress constraints only (total constraints = 15).
- Case 5.* Stress and displacement constraints (total constraints = 21).
- Case 6.* All constraints—stress, displacement, buckling, and frequency (total constraints = 31).

The structure can be analyzed by considering either the equilibrium of node 4 or the general finite element procedures. By following the general procedures, the following expressions for displacements, member stresses, and fundamental vibration frequency are obtained (note that the notations are defined in Section 2.10):

TABLE 14-5 History of the Iterative Process and Final Solution for a Symmetric Three-Bar Structure, Case 2—Stress and Displacement Constraints

<i>I</i>	<i>Max. vio.</i>	<i>Conv. parm.</i>	<i>Cost</i>	$A_1 = A_3$	A_2
1	6.99992E+00	1.00000E+00	3.82843E+00	1.0000E+00	1.0000E+00
2	3.26663E+00	1.00000E+00	6.90598E+00	1.8750E+00	1.6027E+00
⋮	⋮	⋮	⋮	⋮	⋮
8	1.50650E−04	3.05485E−04	2.29695E+01	7.9999E+00	3.4230E−01
9	2.26886E−08	4.53876E−08	2.29704E+01	7.9999E+00	3.4320E−01

Constraint activity

<i>No.</i>	<i>Active</i>	<i>Value</i>	<i>Lagr. mult.</i>
1	Yes	−2.86000E−02	0.00000E+00
2	No	−9.71400E−01	0.00000E+00
3	No	−7.64300E−01	0.00000E+00
4	Yes	1.33227E−15	1.69704E+01
5	Yes	−5.72000E−02	0.00000E+00
6	No	−5.00000E−01	0.00000E+00
7	No	−5.00000E−01	0.00000E+00
8	No	−7.50000E−01	0.00000E+00
9	No	−1.00000E+00	0.00000E+00
10	Yes	2.26886E−08	6.00000E+00
11	No	−9.85700E−01	0.00000E+00
12	No	−5.14300E−01	0.00000E+00
13	No	−8.82150E−01	0.00000E+00
14	No	−5.00000E−01	0.00000E+00
15	No	−5.28600E−01	0.00000E+00

Design variable activity

<i>No.</i>	<i>Active</i>	<i>Design</i>	<i>Lower</i>	<i>Upper</i>	<i>Lagr. mult.</i>
1	No	7.99992E+00	1.00000E−01	1.00000E+02	0.00000E+00
2	No	3.43200E−01	1.00000E−01	1.00000E+02	0.00000E+00

Cost function at optimum = 2.297040E+01

No. of calls for cost function evaluation = 9

No. of calls for evaluation of cost function gradient = 9

No. of calls for constraint function evaluation = 9

No. of calls for evaluation of constraint function gradient = 9

No. of total gradient evaluations = 48

Displacements:

$$u = \frac{l}{E} \left[\frac{(A_1 + 2\sqrt{2}A_2 + A_3)P_u + (A_1 - A_3)P_v}{A_1A_2 + \sqrt{2}A_1A_3 + A_2A_3} \right], \text{ in}$$

$$v = \frac{l}{E} \left[\frac{-(A_1 - A_3)P_u + P_v(A_1 + A_3)}{A_1A_2 + \sqrt{2}A_1A_3 + A_2A_3} \right], \text{ in}$$

TABLE 14-6 History of the Iterative Process and Final Solution for a Symmetric Three-Bar Structure, Case 3—All Constraints

<i>I</i>	<i>Max. vio.</i>	<i>Conv. parm.</i>	<i>Cost</i>	$A_1 = A_3$	A_2
1	6.99992E+00	1.00000E+00	3.82843E+00	1.0000E+00	1.0000E+00
2	2.88848E+00	1.00000E+00	7.29279E+00	2.0573E+00	1.4738E+00
⋮	⋮	⋮	⋮	⋮	⋮
7	7.38741E-05	3.18776E-04	2.29691E+01	7.9993E+00	3.4362E-01
8	5.45657E-09	2.31529E-08	2.29704E+01	7.9999E+00	3.4320E-01

Constraint activity

<i>No.</i>	<i>Active</i>	<i>Value</i>	<i>Lagr. mult.</i>
1	No	-1.56967E-01	0.00000E+00
2	Yes	-2.86000E-02	0.00000E+00
3	No	-9.71400E-01	0.00000E+00
4	No	-7.64300E-01	0.00000E+00
5	Yes	5.45657E-09	1.69704E+01
6	Yes	-5.72000E-02	0.00000E+00
7	No	-5.00000E-01	0.00000E+00
8	No	-5.00000E-01	0.00000E+00
9	No	-7.50000E-01	0.00000E+00
10	No	-1.00000E+00	0.00000E+00
11	Yes	0.00000E+00	6.00000E+00
12	No	-9.85700E-01	0.00000E+00
13	No	-5.14300E-01	0.00000E+00
14	No	-8.82150E-01	0.00000E+00
15	No	-5.00000E-01	0.00000E+00
16	No	-5.28600E-01	0.00000E+00

Design variable activity

<i>No.</i>	<i>Active</i>	<i>Design</i>	<i>Lower</i>	<i>Upper</i>	<i>Lagr. mult.</i>
1	No	7.99992E+00	1.00000E-01	1.00000E+02	0.00000E+00
2	No	3.43200E-01	1.00000E-01	1.00000E+02	0.00000E+00

Cost function at optimum = 2.297040E+01

No. of calls for cost function evaluation = 8

No. of calls for evaluation of cost function gradient = 8

No. of calls for constraint function evaluation = 8

No. of calls for evaluation of constraint function gradient = 8

No. of total gradient evaluations = 50

Member stresses:

$$\sigma_1 = \frac{(\sqrt{2}A_2 + A_3)P_u + A_3P_v}{A_1A_2 + \sqrt{2}A_1A_3 + A_2A_3}, \text{ psi}$$

$$\sigma_2 = \frac{-(A_1 - A_3)P_u + (A_1 + A_3)P_v}{A_1A_2 + \sqrt{2}A_1A_3 + A_2A_3}, \text{ psi}$$

$$\sigma_3 = \frac{-(A_1 + \sqrt{2}A_2)P_u + A_1P_v}{A_1A_2 + \sqrt{2}A_1A_3 + A_2A_3}, \text{ psi}$$

Lowest eigenvalue:

$$\zeta = \frac{3E}{2\sqrt{2}\rho l^2} \left[\frac{A_1 + \sqrt{2}A_2 + A_3 - [(A_1 - A_3)^2 + 2A_2^2]^{1/2}}{\sqrt{2}(A_1 + A_3) + A_2} \right]$$

Fundamental frequency:

$$\omega = \frac{1}{2\pi} \sqrt{\zeta}, \text{ Hz}$$

Tables 14-7 to 14-9 contain the history of the iterative process, constraint, and design variable activities at the final design, and the optimum cost for the three cases with the SQP method in IDESIGN. The active constraints at the optimum point and their Lagrange multipliers (for normalized constraints) are

TABLE 14-7 History of the Iterative Process and Final Solution for an Asymmetric Three-Bar Structure, Case 4—Stress Constraints

<i>I</i>	<i>Max. vio.</i>	<i>Conv. parm.</i>	<i>Cost</i>	<i>A</i> ₁	<i>A</i> ₂	<i>A</i> ₃
1	4.65680E+00	1.00000E+00	3.82843E+00	1.0000E+00	1.0000E+00	1.0000E+00
2	2.10635E+00	1.00000E+00	6.51495E+00	1.9491E+00	1.4289E+00	1.6473E+00
⋮	⋮	⋮	⋮	⋮	⋮	⋮
8	4.03139E-04	2.52483E-03	1.59620E+01	7.0220E+00	2.1322E+00	2.7572E+00
9	4.80986E-07	6.27073E-05	1.59684E+01	7.0236E+00	2.1383E+00	2.7558E+00

Constraint activity

<i>No.</i>	<i>Active</i>	<i>Value</i>	<i>Lagr. mult.</i>
1	Yes	4.80986E-07	1.10020E+01
2	No	-8.38571E-01	0.00000E+00
3	No	-6.45716E-01	0.00000E+00
4	No	-6.57554E-01	0.00000E+00
5	No	-6.96193E-01	0.00000E+00
6	No	-1.27218E-01	0.00000E+00
7	No	-8.22858E-01	0.00000E+00
8	No	-7.94285E-01	0.00000E+00
9	Yes	4.80918E-07	4.96650E+00

Design variable activity

<i>No.</i>	<i>Active</i>	<i>Design</i>	<i>Lower</i>	<i>Upper</i>	<i>Lagr. mult.</i>
1	No	7.02359E+00	1.00000E-01	1.00000E+02	0.00000E+00
2	No	2.13831E+00	1.00000E-01	1.00000E+02	0.00000E+00
3	No	2.75579E+00	1.00000E-01	1.00000E+02	0.00000E+00

Cost function at optimum = 1.596844E+01

No. of calls for cost function evaluation = 9

No. of calls for evaluation of cost function gradient = 9

No. of calls for constraint function evaluation = 9

No. of calls for evaluation of constraint function gradient = 9

No. of total gradient evaluations = 26

TABLE 14-8 History of the Iterative Process and Final Solution for an Asymmetric Three-Bar Structure, Case 5—Stress and Displacement Constraints

<i>I</i>	<i>Max. vio.</i>	<i>Conv. parm.</i>	<i>Cost</i>	<i>A</i> ₁	<i>A</i> ₂	<i>A</i> ₃
1	6.99992E+00	1.00000E+00	3.82843E+00	1.0000E+00	1.0000E+00	1.0000E+00
2	3.26589E+00	1.00000E+00	6.77340E+00	1.9634E+00	1.6469E+00	1.6616E+00
⋮	⋮	⋮	⋮	⋮	⋮	⋮
9	2.18702E-05	3.83028E-04	2.05432E+01	8.9108E+00	1.9299E+00	4.2508E+00
10	6.72142E-09	1.42507E-06	2.05436E+01	8.9106E+00	1.9295E+00	4.2516E+00

Constraint activity

<i>No.</i>	<i>Active</i>	<i>Value</i>	<i>Lagr. Mult.</i>
1	No	-1.95461E-01	0.00000E+00
2	No	-8.47731E-01	0.00000E+00
3	No	-8.04539E-01	0.00000E+00
4	Yes	6.72142E-09	1.19642E+01
5	No	-3.90923E-01	0.00000E+00
6	No	-6.76985E-01	0.00000E+00
7	No	-7.50000E-01	0.00000E+00
8	No	-3.23015E-01	0.00000E+00
9	No	-6.46030E-01	0.00000E+00
10	Yes	6.72141E-09	8.57942E+00
11	No	-9.02269E-01	0.00000E+00
12	No	-8.40435E-01	0.00000E+00
13	No	-2.64008E-01	0.00000E+00
14	No	-1.66277E-01	0.00000E+00
15	No	-3.61739E-01	0.00000E+00

Design variable activity

<i>No.</i>	<i>Active</i>	<i>Design</i>	<i>Lower</i>	<i>Upper</i>	<i>Lagr. mult.</i>
1	No	8.91058E+00	1.00000E-01	1.00000E+02	0.00000E+00
2	No	1.92954E+00	1.00000E-01	1.00000E+02	0.00000E+00
3	No	4.25157E+00	1.00000E-01	1.00000E+02	0.00000E+00

Cost function at optimum = 2.054363E+01

No. of calls for cost function evaluation = 10

No. of calls for evaluation of cost function gradient = 10

No. of calls for constraint function evaluation = 10

No. of calls for evaluation of constraint function gradient = 10

No. of total gradient evaluations = 43

Case 4. Stress in member 1 under loading condition 1, 11.00; stress in member 3 under loading condition 3, 4.97.

Case 5. Horizontal displacement under loading condition 1, 11.96; vertical displacement under loading condition 2, 8.58.

Case 6. Frequency constraint, 6.73; horizontal displacement under loading condition 1, 13.28; vertical displacement under loading condition 2, 7.77.

Note that the optimum weight for Case 5 is higher than that for Case 4, and for Case 6 it is higher than that for Case 5. This is consistent with the previous observation; the number of constraints for Case 5 is larger than that for Case 4, and for Case 6 it is larger than that for Case 5.

TABLE 14-9 History of the Iterative Process and Final Solution for an Asymmetric Three-Bar Structure, Case 6—All Constraints

<i>I</i>	<i>Max. vio.</i>	<i>Conv. parm.</i>	<i>Cost</i>	<i>A</i> ₁	<i>A</i> ₂	<i>A</i> ₃
1	6.99992E+00	1.00000E+00	3.82843E+00	1.0000E+00	1.0000E+00	1.0000E+00
2	2.88848E+00	1.00000E+00	7.29279E+00	2.0573E+00	1.4738E+00	2.0573E+00
⋮	⋮	⋮	⋮	⋮	⋮	⋮
7	6.75406E-05	2.25516E-04	2.10482E+01	8.2901E+00	1.2017E+00	6.7435E+00
8	6.46697E-09	1.88151E-08	2.10494E+01	8.2905E+00	1.2013E+00	5.7442E+00

Constraint activity

<i>No.</i>	<i>Active</i>	<i>Value</i>	<i>Lagr. mult.</i>
1	Yes	2.99788E-09	6.73133E+00
2	No	-1.14125E-01	0.00000E+00
3	No	-8.07062E-01	0.00000E+00
4	No	-8.85875E-01	0.00000E+00
5	Yes	6.46697E-09	1.32773E+01
6	No	-2.28249E-01	0.00000E+00
7	No	-5.90714E-01	0.00000E+00
8	No	-7.50000E-01	0.00000E+00
9	No	-4.09286E-01	0.00000E+00
10	No	-8.18573E-01	0.00000E+00
11	Yes	1.23490E-09	7.77213E+00
12	No	-9.42938E-01	0.00000E+00
13	No	-8.60769E-01	0.00000E+00
14	No	-3.86013E-01	0.00000E+00
15	No	-3.28951E-01	0.00000E+00
16	No	-4.43075E-01	0.00000E+00

Design variable activity

<i>No.</i>	<i>Active</i>	<i>Design</i>	<i>Lower</i>	<i>Upper</i>	<i>Lagr. mult.</i>
1	No	8.29052E+00	1.00000E-01	1.00000E+02	0.00000E+00
2	No	1.20130E+00	1.00000E-01	1.00000E+02	0.00000E+00
3	No	5.74423E+00	1.00000E-01	1.00000E+02	0.00000E+00

Cost function at optimum = 2.104943E+01
 No. of calls for cost function evaluation = 8
 No. of calls for evaluation of cost function gradient = 8
 No. of calls for constraint function evaluation = 8
 No. of calls for evaluation of constraint function gradient = 8
 No. of total gradient evaluations = 48

14.6.3 Comparison of Solutions

Table 14-10 contains a comparison of solutions for all six cases. Since an asymmetric structure has a larger design space, the optimum solutions should be better than those for the symmetric case, and they are; Case 4 is better than Case 1, Case 5 is better than Case 2, and Case 6 is better than Case 3. *These results show that for better practical solutions, more flexibility should be allowed in the design process by defining more design variables; i.e., by allowing more design degrees of freedom.*

TABLE 14-10 Comparison of Optimum Costs for Six Cases of a Three-Bar Structure

	<i>Symmetric structure</i>			<i>Asymmetric structure</i>		
	<i>Case 1</i>	<i>Case 2</i>	<i>Case 3</i>	<i>Case 4</i>	<i>Case 5</i>	<i>Case 6</i>
Optimum weight (lb)	21.11	22.97	22.97	15.97	20.54	21.05
NIT ^a	9	9	8	9	10	8
NCF ^a	9	9	8	9	10	8
NGE ^a	19	48	50	26	43	48

^aNIT, number of iterations; NCF, number of calls for function evaluation; NGE, total number of gradient evaluations.

14.7 Discrete Variable Optimum Design

In many practical applications of optimization, design variables for a problem must be selected from a given set of values. For example, structural elements must be chosen from those that are already commercially available. This is called *discrete variable optimization*, which is essential to economize on fabrication costs for the design. The subject is briefly discussed in Section 2.11.4. We shall demonstrate the procedure described there for a simple design problem.

The application area that we have chosen is the optimum design of aerospace, automotive, mechanical, and structural systems, by employing finite element models. The problem is to design a minimum weight system with constraints on various performance specifications. As a sample application, we shall consider the 10-bar cantilever structure shown in Fig. 14-3. The loading and other design data for the problem are given in Table 14-11. The set of discrete values taken from the American Institute of Steel Construction (AISC) Manual are also given there. The final design for the structure must be selected from this set. The cross-sectional area of each member is treated as a design variable giving a total of 10 variables. Constraints are imposed on member stress (10), nodal displacement (8), member buckling (10), vibration frequency (1), and explicit bounds on the design variables (20). This gives a total of 49 constraints. In imposing the member buckling constraint, the moment of inertia is taken as $I = \beta A^2$, where β is a constant and A is the member cross-sectional area. The formulation for the problem is quite similar to the one for the three-bar structure discussed in Section 2.10. The only difference is that the explicit form of the constraint function is not known. Therefore, we must use the finite element procedures described in Sections 14.1 and 14.2 for structural analysis and the gradient evaluation of constraints.

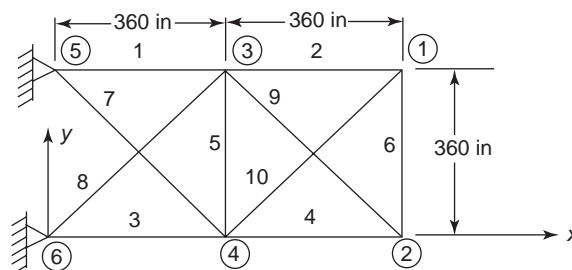
**FIGURE 14-3** Ten-bar cantilever truss.

TABLE 14-11 Design Data for a 10-Bar Structure

Modulus of elasticity:	$E = (1.00E+07)$ psi
Material weight density:	$\gamma = (1.0E-01)$ lb/in ³
Displacement limit:	± 2.0 in
Stress limit:	25,000 psi
Frequency limit:	22 Hz
Lower limit on design variables:	1.62 in ²
Upper limit on design variables:	none
Constant β ($I = \beta A^2$):	1.0

Loading Data:

Node no.	Load in y-direction (lb)
1	50,000
2	-150,000
3	50,000
4	-150,000

Available member sizes (in²): 1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.50, 13.50, 13.90, 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50.

14.7.1 Continuous Variable Optimization

To compare solutions, the continuous variable optimization problem is solved first. To use the program IDESIGN, USER subroutines are developed using the material of Sections 14.1 and 14.2 to evaluate the functions and their gradients. The optimum solution using a very severe convergence criteria and a uniform starting design of 1.62 in², is obtained as

Design variables: 28.28, 1.62, 27.262, 13.737, 1.62, 4.0026, 13.595, 17.544, 19.13, 1.62
 Optimum cost function: 5396.5 lb
 Number of iterations: 19
 Number of analyses: 21
 Maximum constraint violation at optimum: (8.024E-10)
 Convergence parameter at optimum: (2.660E-05)
 Active constraints at optimum and their Lagrange multipliers
 frequency, 392.4
 stress in member 2, 38.06
 displacement at node 2 in the y direction, 4967
 lower bound for member 2, 7.829
 lower bound for member 5, 205.1
 lower bound for member 10, 140.5

14.7.2 Discrete Variable Optimization

We use the adaptive numerical optimization procedure described in Section 2.11.4 to obtain a discrete variable solution. The procedure is to use the program IDESIGN in an interactive mode. Design conditions are monitored and decisions made to fix design variables that are not changing. The interactive facilities used include design variable histories, maximum constraint violation, and the cost function.

Table 14-12 contains a snapshot of the design conditions at various iterations and the decisions made. It can be seen that for the first five iterations the constraint violations are

TABLE 14-12 Interactive Solution for a 10-Member Structure with Discrete Variables

<i>Iteration no.</i>	<i>Maximum violation (%)</i>	<i>Cost function</i>	<i>Algorithm used</i>	<i>Variables fixed to value shown in parentheses</i>
1	1.274E+04	679.83	CC	All free
2	4.556E+03	1019.74	CC	All free
3	1.268E+03	1529.61	CC	All free
4	4.623E+02	2294.42	CC	All free
5	1.144E+02	3441.63	CC	All free
6	2.020E+01	4722.73	CC	5 (1.62), 10 (1.62)
7	2.418E+00	5389.28	CCC	2 (1.80)
11	1.223E-01	5402.62	SQP	1 (30.0), 6 (3.84), 7 (13.5)
13	5.204E-04	5411.13	SQP	3 (26.5), 9 (19.9)
14	1.388E+00	5424.69	—	4 (13.5), 8 (16.9)

CC, constraint correction algorithm; CCC, constraint correction at constant cost; SQP, sequential quadratic programming.

very large, so the constraint correction (CC) algorithm is used to correct the constraints. At the sixth iteration, it is determined that design variables 5 and 10 are not changing, so they are fixed to their current value. Similarly, at other iterations, variables are assigned values from the available set. At the 14th iteration, variables have discrete values, the constraint violation is about 1.4 percent and the structural weight is 5424.69, which is an increase of less than 1 percent from the true optimum. This is a reasonable final solution.

It should be noted that with the discrete variables, several solutions near the true optimum point are possible. A different sequence of fixing variables can give a different solution. For example, starting from the optimum solution with continuous variables, the following acceptable discrete solutions are obtained interactively:

1. 30.0, 1.62, 26.5, 13.9, 1.62, 4.18, 13.5, 18.8, 18.8, 1.62; cost = 5485.6, max. viol. = 4.167 percent for stress in member 2.
2. Same as (1) except the eighth design variable is 16.9; cost = 5388.9 and max. viol. = 0.58 percent.
3. Same as (1) except design variables 2 and 6 are 2.38 and 2.62; cost = 5456.8, max. viol. = 3.74 percent for stress in member 2
4. Same as (3) except design variable 2 is 2.62; cost = 5465.4; all constraints are satisfied.

It can be seen that the interactive facilities described in Chapter 13 can be exploited to obtain practical engineering designs.

14.8 Optimal Control of Systems by Nonlinear Programming

14.8.1 A Prototype Optimal Control Problem

Optimal control problems are dynamic in nature. A brief discussion of differences between optimal control and optimum design problems is given there. It turns out that some optimal control problems can be formulated and solved by the nonlinear programming methods described in Chapters 11 and 13. In this section, we consider a simple optimal control problem that has numerous practical applications. Various formulations of the problem are described and optimal solutions are obtained and discussed.

The application area that we have chosen to demonstrate the use of nonlinear programming methods for this class of problems is the vibration control of systems. This is an important class of problems that is encountered in numerous real-world applications. For example, the control of structures under earthquake and wind loads, vibration control of sensitive instruments to blast loading or shock input, control of the large space structures, precision control of machines, among others. To treat these problems we shall consider a simple model of the system to demonstrate the basic formulation and solution procedure. Using the demonstrated procedures, more complex models can be treated to simulate the real-world systems more accurately.

To treat optimal control problems, dynamic response analysis capability must be available. In the present text, we shall assume that students have some background in vibration analysis of systems. In particular, we shall model systems as single-degree-of-freedom linear spring-mass systems. This leads to a second-order linear differential equation whose closed-form solution is available (Clough and Penzien, 1975; Chopra, 1995). It may be worthwhile for the students to briefly review the material on the solution of linear differential equations.

To demonstrate the formulation and the solution process, we consider a cantilever structure shown in Fig. 14-4. The data for the problem and various notations used in Fig. 14-4 are defined in Table 14-13. The structure is a highly idealized model of many systems that are used in practice. The length of the structure is L and its cross section is rectangular with width as b and depth as h . The system is at rest initially at time $t = 0$. It experiences a sudden load due to a shock wave or other similar causes. The problem is to control the vibrations of the system such that the displacements are not too large and the system comes to rest in a controlled manner. The system has proper sensors and actuators that generate the desired force to suppress the vibrations and bring the system to rest. The control force may also be generated by properly designed dampers or viscoelastic support pads along the length of the structure. We shall not discuss the detailed design of the control force generating mechanisms, but we shall discuss the problem of determining the optimum shape of the control force.

The governing equation that describes the motion of the system is a second-order partial differential equation. To simplify the analysis, we use separation of variables, and express the deflection function $y(x, t)$ as

$$y(x, t) = \psi(x)q(t) \quad (14.26)$$

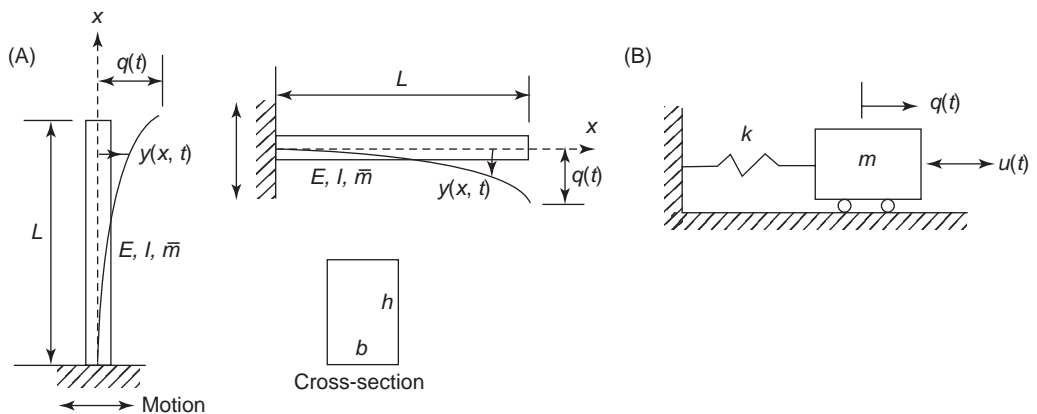


FIGURE 14-4 Model of a system subjected to shock input. (A) Cantilever structures subjected to shock input at support. (B) Equivalent single degree of freedom model.

TABLE 14-13 Data for the Optimal Control Problem

Length of the structure:	$L = 1.0 \text{ m}$
Width of cross section:	$b = 0.01 \text{ m}$
Depth of cross section:	$h = 0.02 \text{ m}$
Modulus of elasticity:	$E = 200 \text{ GPa}$
Mass density:	$\rho = 7800 \text{ kg/m}^3$
Moment of inertia:	$I = (6.667\text{E}-09) \text{ m}^4$
Mass per unit length:	$\bar{m} = 1.56 \text{ kg/m}$
Control function:	$u(t) = \text{to be determined}$
Limit on the control function:	$u_a = 30 \text{ N}$
Initial velocity:	$v_0 = 1.5 \text{ m/s}$

where $\psi(x)$ is a known function called the shape function, and $q(t)$ is the displacement at the tip of the cantilever, as shown in Fig. 14-4. Several shape functions can be used; however, we shall use the following one:

$$\psi(x) = \frac{1}{2}(3\xi^2 - \xi^3); \quad \xi = \frac{x}{L} \quad (14.27)$$

Using kinetic and potential energies for the system, $\psi(x)$ of Eq. (14.27), and the data of Table 14-13, the mass and spring constants for an equivalent single-degree-of-freedom system shown in Fig. 14-4 are calculated as follows (Clough and Penzien, 1975; Chopra, 1995):

Mass

$$\begin{aligned} \text{kinetic energy} &= \frac{1}{2} \int_0^L \bar{m} \dot{y}^2(t) dx \\ &= \frac{1}{2} \left[\int_0^L \bar{m} \psi^2(x) dx \right] \dot{q}^2(t) \\ &= \frac{1}{2} m \dot{q}^2(t) \end{aligned}$$

where the mass m is identified as

$$\begin{aligned} m &= \int_0^L \bar{m} \psi^2(x) dx \\ &= \frac{33}{140} \bar{m} L \\ &= \frac{33}{140} (1.56)(1.0) = 0.3677 \text{ kg} \end{aligned}$$

Spring constant

$$\begin{aligned} \text{strain energy} &= \frac{1}{2} \int_0^L EI [y''(x)]^2 dx \\ &= \frac{1}{2} \left[\int_0^L EI (\psi''(x))^2 dx \right] q^2(t) \\ &= \frac{1}{2} k q^2(t) \end{aligned}$$

where the spring constant k is identified as

$$\begin{aligned} k &= \int_0^L EI(\psi''(x))^2 dx \\ &= \frac{3EI}{L^3} \\ &= 3(2.0\text{E}+11)(6.667\text{E}-09)/(1.0)^3 = 4000 \text{ N/m} \end{aligned}$$

In the foregoing, a “dot” over a variable indicates derivatives with respect to time and a “prime” indicates derivatives with respect to the coordinate x .

The equation, of motion for the single-degree-of-freedom system along with the initial conditions (initial displacement q_0 , initial velocity v_0) are given as

$$m\ddot{q}(t) + kq(t) = u(t) \quad (14.28)$$

$$q(0) = q_0, \quad \dot{q}(0) = v_0 \quad (14.29)$$

where $u(t)$ is the control force needed to suppress vibrations due to the initial velocity v_0 (*shock loading* for the system is transformed to an equivalent initial velocity calculated as impulse of the force divided by the mass). Note that the material damping for the system is neglected. Therefore, if no control force $u(t)$ is used, the system will continue to oscillate. Figures 14-5 and 14-6 show the displacement and velocity response of the system for the initial 0.10s when $u(t) = 0$, i.e., no control mechanism is used.

The control problem is to determine the forcing function $u(t)$ such that the system comes to rest in a specified time. We can also pose the problem as follows: Determine the control force to minimize the time to bring the system to rest. We shall investigate several of these formulations in the following paragraphs.

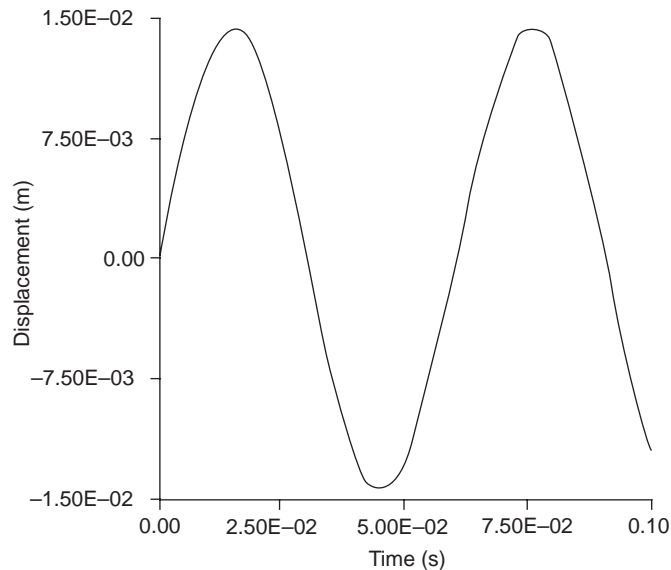


FIGURE 14-5 Displacement response of the equivalent single-degree-of-freedom system to shock loading with no control force.

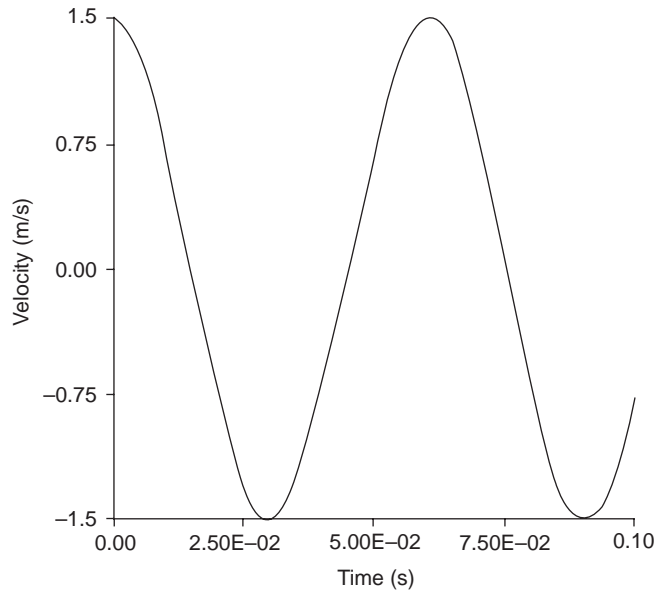


FIGURE 14-6 Velocity response of the equivalent single-degree-of-freedom system to shock loading with no control force.

We note here that for the preceding simple problem, solution procedures other than the nonlinear programming methods are available (Meirovitch, 1985). Those procedures may be better for the simple problem. However, we shall use nonlinear programming formulations to solve the problem to demonstrate generality of the method.

14.8.2 Minimization of Error in State Variable

As a first formulation, we define the performance index (cost function) as minimization of error in the state variable (response) in the time interval 0 to T as

$$f_1 = \int_0^T q^2(t) dt \quad (14.30)$$

Constraints are imposed on the terminal response, the displacement response, and the control force as follows:

$$\text{Displacement constraint: } |q(t)| \leq q_a \text{ in the time interval } 0 \text{ to } T \quad (14.31)$$

$$\text{Terminal displacement constraint: } q(T) = q_T \quad (14.32)$$

$$\text{Terminal velocity constraint: } \dot{q}(T) = v_T \quad (14.33)$$

$$\text{Control force constraint: } |u(t)| \leq u_a \text{ in the interval } 0 \text{ to } T \quad (14.34)$$

where q_a is the maximum allowed displacement of the system, q_T and v_T are small specified constants, and u_a is the limit on the control force. Thus, the design problem is to compute the control function $u(t)$ in the time interval 0 to T to minimize the performance index of Eq. (14.30) subject to the constraints of Eqs. (14.31) to (14.34) and satisfaction of the equations of motion (14.28) and the initial conditions in Eq. (14.29). Note that the constraints of Eqs. (14.31) and (14.34) are dynamic in nature and need to be satisfied over the entire time interval 0 to T .

Another performance index can be defined as the sum of the squares of the displacement and the velocity as:

$$f_2 = \int_0^T [q^2(t) + \dot{q}^2(t)] dt \quad (14.35)$$

Formulation for Numerical Solution In order to obtain numerical results, the following data are used:

Allowable time to suppress motion:	$T = 0.10$ s
Initial velocity:	$v_0 = 1.5$ m/s
Initial displacement:	$q_0 = 0.0$ m
Allowable displacement:	$q_a = 0.01$ m
Terminal velocity:	$v_T = 0.0$ m/s
Terminal displacement:	$q_T = 0.0$ m
Limit on the control force:	$u_a = 30.0$ N

For the present example, the equation of motion is quite simple, and its analytical solution can be written using Duhamel's integral (Clough and Penzien, 1975; Chopra, 1995) as follows:

$$q(t) = \frac{1}{\omega} v_0 \sin \omega t + q_0 \cos \omega t + \frac{1}{m\omega} \int_0^t u(\eta) \sin \omega(t - \eta) d\eta \quad (14.36)$$

$$\dot{q}(t) = v_0 \cos \omega t - q_0 \omega \sin \omega t + \frac{1}{m} \int_0^t u(\eta) \cos \omega(t - \eta) d\eta \quad (14.37)$$

In more complex applications, the equations of motion will have to be integrated using numerical methods (Shampine and Gordon, 1975; Hsieh and Arora, 1984).

Since explicit forms for the displacement and velocity in terms of the design variable $u(t)$ are known, we can calculate their derivatives explicitly by differentiating Eqs. (14.36) and (14.37) with respect to $u(\eta)$, where η is a point between 0 and T , as

$$\begin{aligned} \frac{dq(t)}{du(\eta)} &= \frac{1}{m\omega} \sin \omega(t - \eta) \quad \text{for } t \geq \eta \\ &= 0 \quad \text{for } t < \eta \end{aligned} \quad (14.38)$$

$$\begin{aligned} \frac{d\dot{q}(t)}{du(\eta)} &= \frac{1}{m} \cos \omega(t - \eta) \quad \text{for } t \geq \eta \\ &= 0 \quad \text{for } t < \eta \end{aligned} \quad (14.39)$$

In the foregoing expressions, $du(t)/du(\eta) = \delta(t - \eta)$ has been used, where $\delta(t - \eta)$ is the Dirac delta function. The derivative expressions can easily be programmed to impose constraints on the problem. For more general applications, derivatives must be evaluated using numerical computational procedures. Several such procedures developed and evaluated by Hsieh and Arora (1984) and Tseng and Arora (1987) can be used for more complex applications.

Equations (14.36) to (14.39) are used to develop the user-supplied subroutines for the program IDESIGN. Several procedures are needed to solve the problem numerically. First of all, a grid must be used to discretize the time where displacement, velocity, and the control

force are evaluated. Interpolation methods, such as cubic splines, B-splines (De Boor, 1978), etc., can be used to evaluate the functions at points other than the grid points.

Another difficulty concerns the dynamic displacement constraint of Eq. (14.31). The constraint must be imposed during the entire time interval 0 to T . Several treatments for such constraints have been investigated (Hsieh and Arora, 1984; Tseng and Arora, 1987). For example, the constraint can be replaced by several constraints imposed at the local maximum points for the function $q(t)$; it may be replaced by an integral constraint; or it may be imposed at each grid point.

In addition to the foregoing numerical procedures, a numerical integration scheme, such as simple summation, trapezoidal rule, Simpson's rule, Gaussian quadrature, and so on, must be selected for evaluating the integrals in Eqs. (14.30), (14.36), and (14.37). Based on some preliminary investigations, the following numerical procedures are selected for their simplicity to solve the present problem:

Numerical integration:	Simpson's method
Dynamic constraint:	imposed at each grid point
Design variable (control force):	value at each grid point

Numerical Results Using the foregoing procedures and the numerical data, the problem is solved using the SQP method of Section 11.4 available in the IDESIGN software package. The number of grid points is selected as 41, so there are 41 design variables. The displacement constraint of Eq. (14.31) is imposed at the grid points with its limit set as $q_a = 0.01$ m. As an initial estimate, $u(t)$ is set to zero, so constraints of Eqs. (14.31) to (14.33) are violated. The algorithm finds a feasible design in just three iterations. During these iterations, the cost function of Eq. (14.30) is also reduced. The algorithm reaches near to the optimum point at the 11th iteration. As a result of the severe convergence criteria, it takes another 27 iterations to satisfy the specified convergence criteria. The cost function history is plotted in Fig. 14-7. For all practical purposes the optimum solution is obtained somewhere between

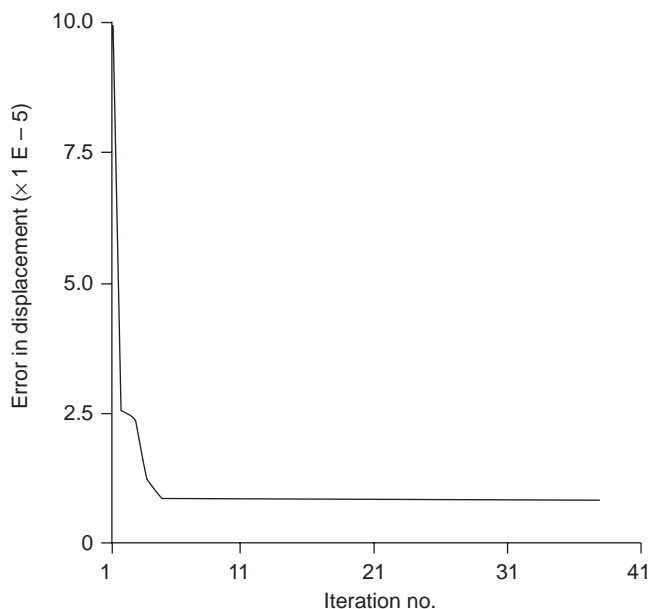


FIGURE 14-7 Cost function history for the optimal control problem of minimizing the error in the state variable (cost function f_1).

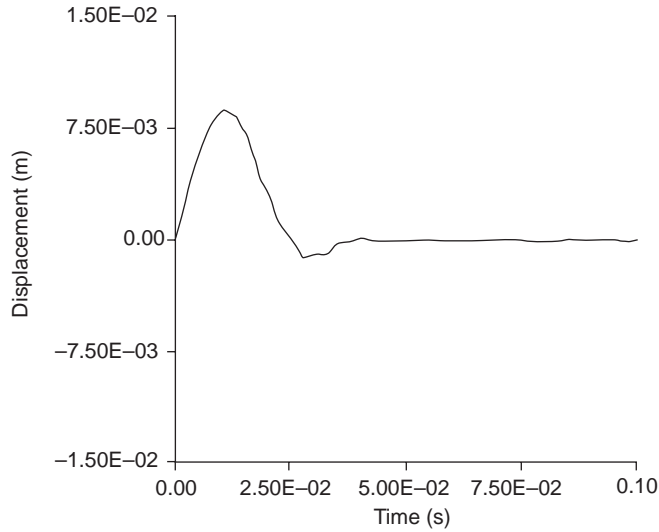


FIGURE 14-8 Displacement response at optimum with minimization of error in the state variable as the performance index (cost function f_1).

the 15th and 30th iterations. Thus, if IDESIGN was being executed in the interactive mode, the designer could terminate the iterative process at the 17th iteration.

The final displacement and velocity responses and the control force history are shown in Figs. 14-8 to 14-10. It can be observed that the displacement and velocity both go to zero at about 0.05 s, so the system comes to rest at that point. The control force also has zero value after that point and reaches its limit value at several points during that interval. The final cost function value is (8.536E-07).

Effect of Problem Normalization It turns out that for the present application it is advantageous to normalize the problem and optimize it with normalized variables. We shall briefly discuss these normalizations which can also be useful in other applications. Without normalization of the present problem, the cost function and its gradient as well as constraint functions and their gradients have quite small values. The algorithm required a very small value for the convergence parameter (1.0E-09) to converge to the same optimum solution as with the normalized problem. In addition, the rate of convergence without normalization was also quite slow. This apparent numerical difficulty was due to ill-conditioning in the problem which was overcome by the normalization procedure that is described in the following.

The independent variable transformation for the time is defined as

$$t = \tau T \quad \text{or} \quad \tau = \frac{t}{T} \quad (14.40)$$

where τ is the normalized independent variable. With this transformation, when t varies between 0 and T , τ varies between 0 and 1. The displacement is normalized as

$$q(t) = Tq_{\max}\bar{q}(\tau) \quad \text{or} \quad \bar{q}(\tau) = \frac{q(t)}{Tq_{\max}} \quad (14.41)$$

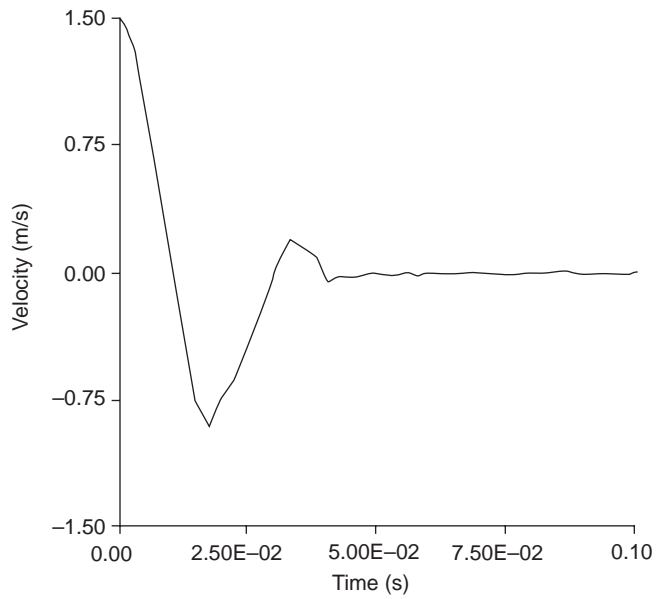


FIGURE 14-9 Velocity response at optimum with minimization of error in the state variable as the performance index (cost function f_1).

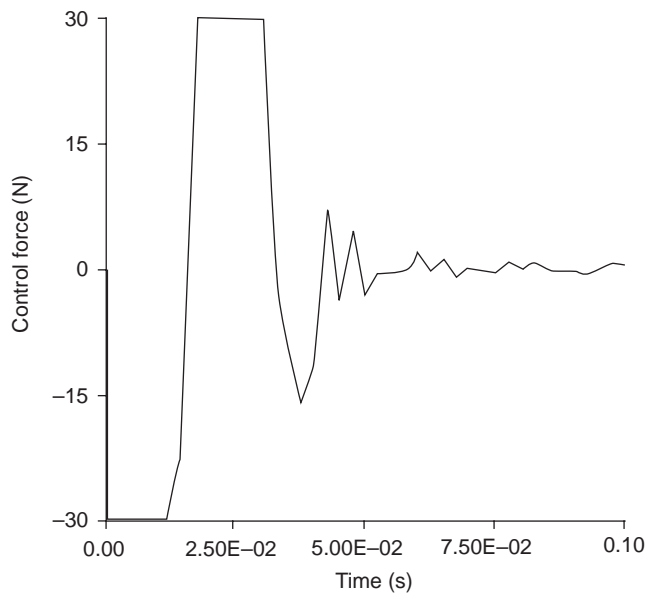


FIGURE 14-10 Optimum control force to minimize error in the state variable due to shock input (cost function f_1).

where $\bar{q}(\tau)$ is the normalized displacement and q_{\max} is taken as 0.015. Derivatives of the displacement with respect to time are transformed as

$$\dot{q}(t) = q_{\max} \dot{\bar{q}}(\tau) \quad (14.42)$$

$$\dot{q}(0) = q_{\max} \dot{\bar{q}}(0) \quad \text{or} \quad \dot{\bar{q}}(0) = \frac{v_0}{q_{\max}} \quad (14.43)$$

$$\ddot{q}(t) = \frac{1}{T} q_{\max} \ddot{\bar{q}}(\tau) \quad (14.44)$$

The control force is normalized as

$$u(t) = u_{\max} \bar{u}(\tau), \quad \text{or} \quad \bar{u}(\tau) = \frac{u(t)}{u_{\max}} \quad (14.45)$$

With this normalization, $\bar{u}(\tau)$ varies between -1 and 1 as $u(t)$ varies between $-u_{\max}$ and u_{\max} . Substituting the preceding transformations into Eqs. (14.28) and (14.29), we get

$$\bar{m} \ddot{\bar{q}}(\tau) + \bar{k} \bar{q}(\tau) = \bar{u}(\tau) \quad (14.46)$$

$$\bar{q}(0) = \frac{q_0}{T q_{\max}}, \quad \dot{\bar{q}}(0) = \frac{v_0}{q_{\max}} \quad (14.47)$$

$$\bar{k} = \frac{kT}{u_{\max}} q_{\max}, \quad \bar{m} = \frac{m}{T u_{\max}} q_{\max}$$

The constraints of Eqs. (14.31) to (14.34) are also normalized as

$$\text{Displacement constraint:} \quad |\bar{q}(\tau)| \leq \frac{q_0}{T q_{\max}} \quad \text{in the interval } 0 \leq \tau \leq 1 \quad (14.48)$$

$$\text{Terminal displacement constraint:} \quad \bar{q}(1) = \frac{1}{T q_{\max}} q_T \quad (14.49)$$

$$\text{Terminal velocity constraint:} \quad \dot{\bar{q}}(1) = \frac{1}{q_{\max}} v_T \quad (14.50)$$

$$\text{Control force constraint:} \quad |\bar{u}(\tau)| \leq 1 \quad \text{in the interval } 0 \leq \tau \leq 1 \quad (14.51)$$

With the foregoing normalizations, the numerical algorithm behaved considerably better and convergence to the optimum solution as reported earlier was quite rapid. Therefore, for general usage, normalization of the problem is recommended whenever possible. Note that many forms of normalizations of a problem are possible. If one form does not work, others should be tried. We shall use the foregoing normalizations in the two formulations discussed in Sections 14.8.3 and 14.8.4.

Discussion of Results The final solution for the problem can be affected by the number of grid points and the convergence criterion. The solution reported previously was obtained using 41 grid points and a convergence criterion of (1.0E-03). A stricter convergence criterion of (1.0E-06) also gave the same solution, using a few more iterations.

The number of grid points can also affect the accuracy of the final solution. The use of 21 grid points also gave approximately the same solution. The shape of the final control force was slightly different. The final cost function value was slightly higher than that with 41 grid points, as expected.

It is also important to note that the problem can become infeasible if the limit q_a on the displacement in Eq. (14.31) is too severe. For example, when q_a was set to 0.008 m, the problem was infeasible with 41 grid points. However, with 21 grid points a solution was obtained. This also shows that when the number of grid points is small, the displacement constraint may actually be violated between the grid points, although it is satisfied at the grid points. Therefore, the number of grid points should be selected judiciously.

The foregoing discussion shows that to impose the constraints more precisely, the exact local max-points should be located and the constraint imposed there. To locate the exact max-points, interpolation procedures may be used, or bisection of the interval in which the max-point lies can be used (Hsieh and Arora, 1984). The gradient of the constraint must be evaluated at the max-points. For the present problem, the preceding procedure is not too difficult to implement because the analytical form for the response is known. For more general applications, the computational as well as programming effort can increase substantially to implement the foregoing procedure.

It is worthwhile to note that several other starting points for the control force such as $u(t) = -30\text{N}$, $u(t) = 30\text{N}$, converged to the same solution as given in Figs. 14-8 to 14-10. The computational effort varied somewhat. The CPU time with 21 grid points was about 20 percent of that with 41 grid points when $u(t) = 0$ was used as the starting point.

It is interesting to note that at the optimum, the dynamic constraint of Eq. (14.31) is not active at any time grid point. It is violated at many intermediate iterations. Also, the terminal response constraints of Eqs. (14.32) and (14.33) are satisfied at the optimum with normalized Lagrange multipliers as $(-7.97\text{E}-04)$ and $(5.51\text{E}-05)$. Since the multipliers are almost zero, the constraints can be somewhat relaxed without affecting the optimal solution. This can be observed from the final displacement and velocity responses shown in Figs. 14-8 and 14-9, respectively. Since the system is essentially at rest after $t = 0.05\text{ s}$, there is no effect of imposing the terminal constraints of Eqs. (14.32) and (14.33).

The control force is at its limit value ($u_a = 30\text{N}$) at several grid points; for example, it is at its lower limit at the first six grid points and at the upper limit at the next six. The Lagrange multiplier for the constraint has its largest value initially and gradually decreases to zero after the 13th grid point. According to the Constraint Variation Sensitivity Theorem 4.7, the optimum cost function can be reduced substantially if the limit on the control force is relaxed for a small duration after the system is disturbed.

14.8.3 Minimum Control Effort Problem

Another formulation for the problem is possible where we minimize the total control effort calculated as

$$f_3 = \int_0^T u^2(t) dt \quad (14.52)$$

The constraints are the same as defined in Eqs. (14.31) to (14.34) and Eqs. (14.28) and (14.29). The numerical procedures for obtaining an optimum solution for the problem are the same as described previously in Section 14.8.2.

This formulation of the problem is quite well-behaved. The same optimum solution is obtained quite rapidly (9–27 iterations) with many different starting points. Figures 14-11 to 14-14 give the cost function history, displacement and velocity responses, and the control force at the optimum solution, which is obtained by starting from $u(t) = 0$ and 41 grid points. The final control effort of 7.481 is much smaller than that for the first case where it was 28.74. The system, however, comes to rest at 0.10 s compared with 0.05 s in the previous case. The solution with 21 grid points resulted in a slightly smaller control effort as a result of the numerical procedures used, as explained earlier.

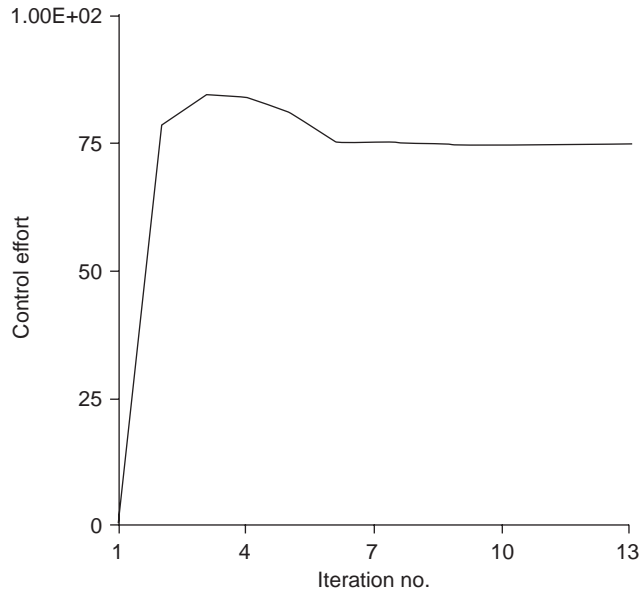


FIGURE 14-11 Cost function history for the optimal control problem of minimization of the control effort (cost function f_3).

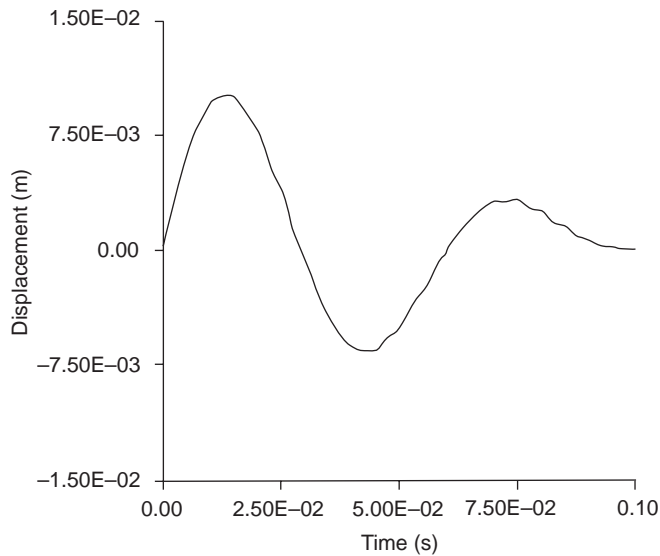


FIGURE 14-12 Displacement response at optimum with minimization of control effort as the performance index (cost function f_3).

It is interesting to note that the constraint of Eq. (14.31) is active at the eighth grid point with the normalized Lagrange multiplier as $(2.429E-02)$. The constraints of Eqs. (14.32) and (14.33) are also active with the normalized Lagrange multipliers as $(-1.040E-02)$ and $(-3.753E-04)$. In addition, the control force is at its lower limit at the first grid point with the Lagrange multiplier as $(7.153E-04)$. This shows that by increasing or decreasing the limit on the control force, the optimum cost function will not be affected significantly.

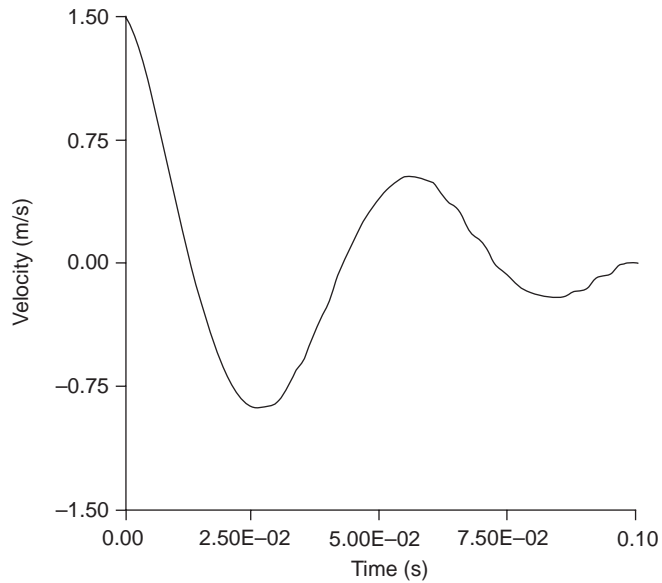


FIGURE 14-13 Velocity response at optimum with minimization of control effort as the performance index (cost function f_3).

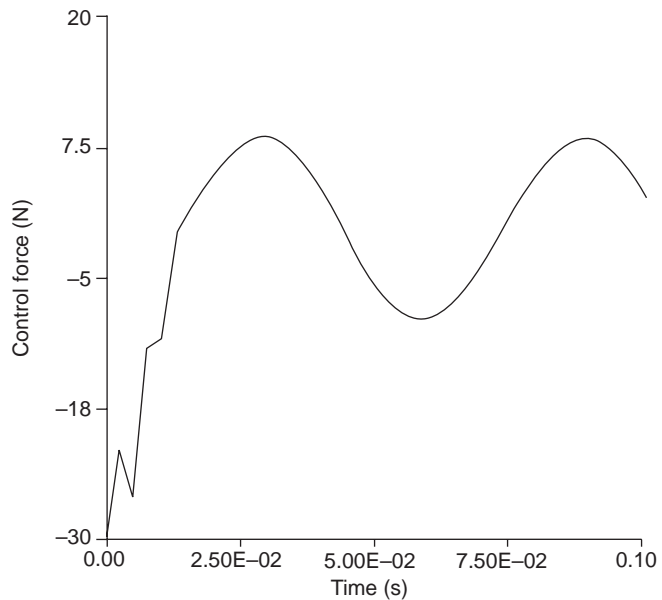


FIGURE 14-14 Optimum control force to minimize the control effort to bring the system to rest after shock input (cost function f_3).

14.8.4 Minimum Time Control Problem

The idea of this formulation is to minimize the time required to suppress the motion of the system subject to various constraints. In the previous formulations, the desired time to bring the system to rest was specified. In the present formulation, however, we try to minimize the time T . Therefore, the cost function is

$$f_4 = T \quad (14.53)$$

The constraints on the system are the same as defined in Eqs. (14.28), (14.29), and (14.31) to (14.34). Note that compared with the previous formulations, gradients of constraints with respect to T are also needed. They can be computed quite easily, since analytical expressions for the functions are known.

The same optimum solution is obtained by starting from several points, such as $T = 0.1, 0.04, 0.02$, and $u(t) = 0, 30, -30$. Figures 14-15 to 14-18 show the cost function history, the

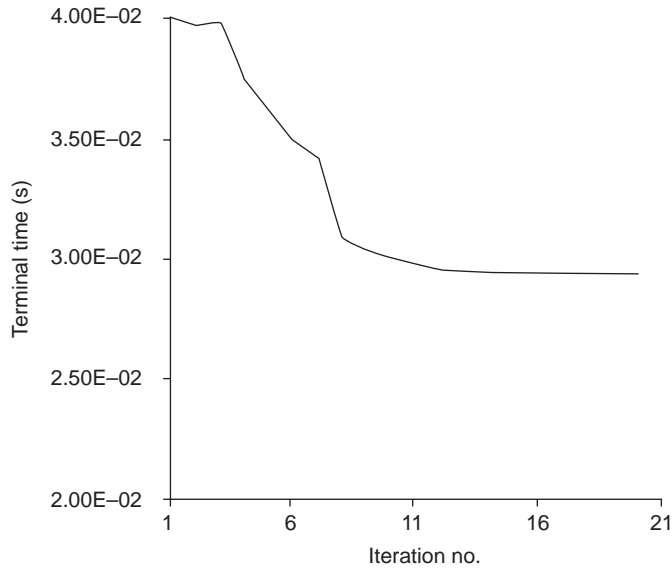


FIGURE 14-15 Cost function history for the optimal control problem of minimization of time to bring the system to rest (cost function f_4).

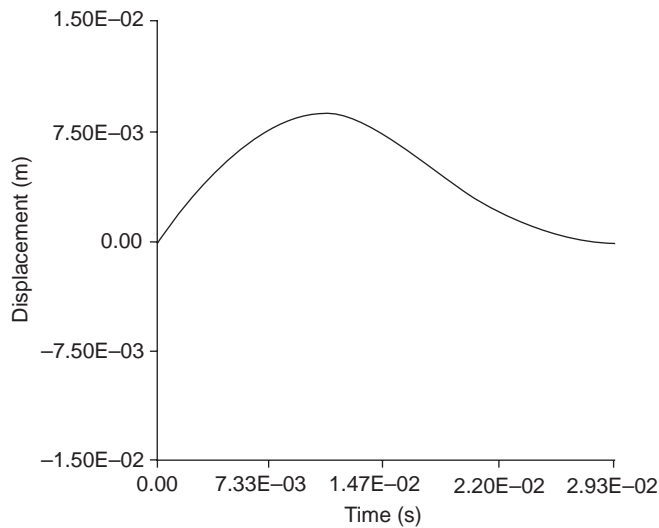


FIGURE 14-16 Displacement response at optimum with minimization of time as the performance index (cost function f_4).

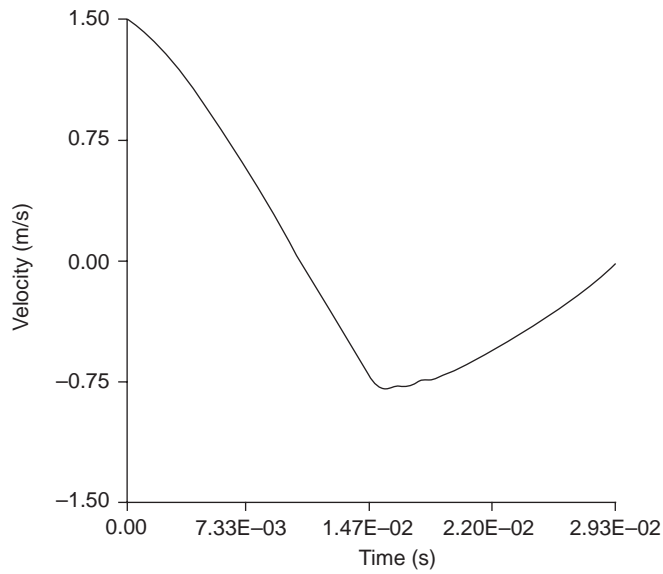


FIGURE 14-17 Velocity response at optimum with minimization of time as the performance index (cost function f_4).

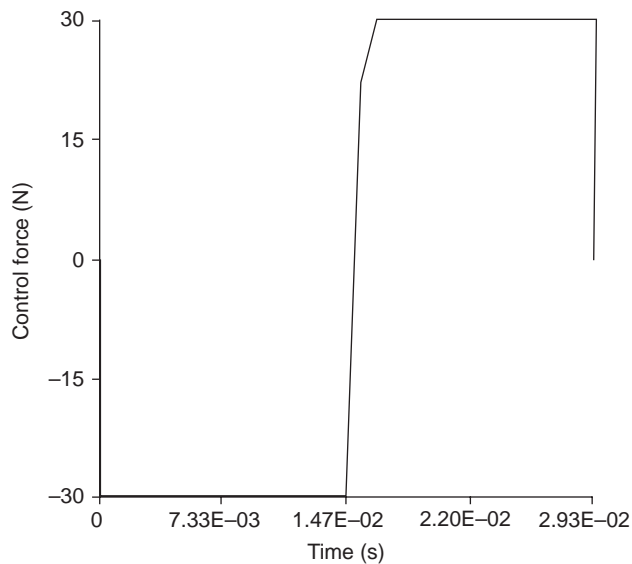


FIGURE 14-18 Optimum control force to minimize time to bring the system to rest after shock input (cost function f_4).

displacement and velocity responses, and the control force at the optimum with 41 grid points, $T = 0.04$, and $u(t) = 0$ as the starting point. The CPU time for the entire solution process is 302 s on the Apollo DN460 workstation. It takes 0.02933 s to bring the system to rest. Depending on the starting point, the number of iterations to converge to the final solution varies between 20 and 56.

TABLE 14-14 Summary of Optimum Solutions for Three Formulations of Optimal Control of Motion of a System Subjected to Shock Input

	<i>Formulation 1: Minimization of error in state variable</i>	<i>Formulation 2: Minimization of control effort</i>	<i>Formulation 3: Minimization of terminal time</i>
f_1	8.53607E-07	2.32008E-06	8.64466E-07
f_2	1.68241E-02	2.73540E-02	1.45966E-02
f_3	2.87398E+01	7.48104	2.59761E+01
f_4	0.10	0.10	2.9336E-02
NIT	38	13	20
NCF	38	13	20
NGE	100	68	64
CPU	212	43	302

NIT, number of iterations; NCF, number of calls for function evaluation; NGE, total number of gradient evaluations.

Constraints of Eqs. (14.32) and (14.33) are active with the normalized Lagrange multipliers as (6.395E-02) and (-1.771E-01), respectively. The control force is at its lower limit for the first 22 grid points and at the upper limit at the remaining points.

Interestingly, even after normalization of the problem, the rate of convergence for the problem could be affected by multiplying the cost function by a positive constant. For example, the number of iterations decreased from 60 to 20 when a factor of 1000 was used. The results reported in the preceding discussion use a factor of 1000. Apparently, the use of this factor affects the step size determination process for the algorithm. One conclusion that can be drawn based on the study is that the step size determination process has room for further improvement in the IDESIGN system.

14.8.5 Comparison of Three Formulations for the Optimal Control of System Motion

It is interesting to compare the three formulations for the optimal control of motion of the system shown in Fig. 14-4. Table 14-14 contains a summary of the optimum solutions with the three formulations. All the solutions are obtained with 41 grid points and $u(t) = 0$ as the starting point using an Apollo DN460 workstation. For the third formulation, $T = 0.04$ s is used as the starting point.

The results of Table 14-14 show that the control effort is largest with the first formulation and the smallest with the second one. The second formulation turns out to be the most efficient as well as convenient to implement. By varying the total time T , this formulation can be used to generate results for Formulation 3. For example, using $T = 0.05$ and 0.02933 s, solutions with Formulation 2 were obtained. With $T = 0.02933$ s, the same results as with Formulation 3 were obtained. Also, when $T = 0.025$ s was used, Formulation 2 resulted in an infeasible problem. For practical applications, Formulation 2 is recommended for the vibration control problems.

Exercises for Chapter 14*

Formulate and solve the following design problems using a nonlinear programming algorithm starting with a reasonable design estimate. Also solve the problems graphically whenever possible and trace the history of the iterative process on the graph of the problem.

- 14.1 Exercise 3.34 14.2 Exercise 3.35 14.3 Exercise 3.36
- 14.4 Exercise 3.50 14.5 Exercise 3.51 14.6 Exercise 3.52
- 14.7 Exercise 3.53 14.8 Exercise 3.54 14.9 Exercise 12.9
- 14.10 Exercise 12.10 14.11 Exercise 12.11 14.12 Exercise 12.12
- 14.13 *Design of a tapered flag pole.* Formulate the flag pole design problem of Exercise 3.52 for the data given there. Use a hollow tapered circular tube with constant thickness as the structural member. The mass of the pole is to be minimized subject to various constraints. Use a numerical optimization method to obtain the final solution and compare it with the optimum solution for the uniform flag pole.
- 14.14 *Design of a sign support.* Formulate the sign support column design problem described in Exercise 3.53 for the data given there. Use a hollow tapered circular tube with constant thickness as the structural member. The mass of the pole is to be minimized subject to various constraints. Use a numerical optimization method to obtain the final solution and compare it with the optimum solution for the uniform column.
- 14.15 Repeat the problem of Exercise 14.13 for a hollow square tapered column of uniform thickness.
- 14.16 Repeat the problem of Exercise 14.14 for a hollow square tapered column of uniform thickness.
- 14.17 For the optimal control problem of minimization of error in the state variable formulated and solved in Section 14.8.2, study the effect of changing the limit on the control force (u_a) to 25 N or 35 N.
- 14.18 For the minimum control effort problem formulated and solved in Section 14.8.3, study the effect of changing the limit on the control force (u_a) to 25 N or 35 N.
- 14.19 For the minimum time control problem formulated and solved in Section 14.8.4, study the effect of changing the limit on the control force (u_a) to 25 N or 35 N.
- 14.20 For the optimal control problem of minimization of error in the state variable formulated and solved in Section 14.8.2, study the effect of having an additional lumped mass M at the tip of the beam ($M = 0.05$ kg) as shown in Fig. E14-20.
- 14.21 For the minimum control effort problem formulated and solved in Section 14.8.3, study the effect of having an additional mass M at the tip of the beam ($M = 0.05$ kg).
- 14.22 For the minimum time control problem formulated and solved in Section 14.8.4, study the effect of having an additional lumped mass M at the tip of the beam ($M = 0.05$ kg).
- 14.23 For Exercise 14.20, what will be the optimum solution if the tip mass M is treated as a design variable with limits on it as $0 \leq M \leq 0.10$ kg?
- 14.24 For Exercise 14.21, what will be the optimum solution if the tip mass M is treated as a design variable with limits on it as $0 \leq M \leq 0.10$ kg?
- 14.25 For Exercise 14.22, what will be the optimum solution if the tip mass M is treated as a design variable with limits on it as $0 \leq M \leq 0.10$ kg?

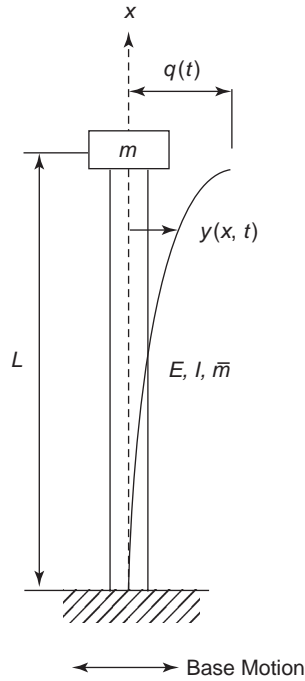


FIGURE E14-20 Cantilever structure with mass at the tip.

- 14.26 For the optimal control problem of minimization of error in the state variable formulated and solved in Section 14.8.2 study the effect of including a 1 percent critical damping in the formulation.
- 14.27 For the minimum control effort problem formulated and solved in Section 14.8.3, study the effect of including a 1 percent critical damping in the formulation.
- 14.28 For the minimum time control problem formulated and solved in Section 14.8.4, study the effect of including a 1 percent critical damping in the formulation.
- 14.29 For the spring-mass-damper system shown in Fig. E14-29, formulate and solve the problem of determining the spring constant and damping coefficient to minimize the maximum acceleration of the system over a period of 10s when it is subjected to an initial velocity of 5 m/s. The mass is specified as 5 kg.

The displacement of the mass should not exceed 5 cm for the entire time interval of 10s. The spring constant and the damping coefficient must also remain within the

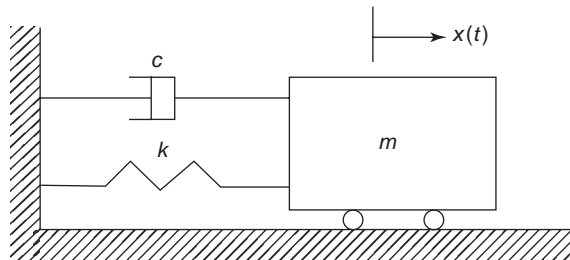


FIGURE E14-29 Damped single-degree-of-freedom system.

limits $1000 \leq k \leq 3000 \text{ N/m}$; $0 \leq c \leq 300 \text{ N}\cdot\text{S/m}$. (*Hint*: The objective of minimizing the maximum acceleration is a min–max problem, which can be converted to a nonlinear programming problem by introducing an artificial design variable. Let $a(t)$ be the acceleration and A be the artificial variable. Then the objective can be to minimize A subject to an additional constraint $|a(t)| \leq A$ for $0 \leq t \leq 10$).

- 14.30 Formulate the problem of optimum design of steel transmission poles described in Kocer and Arora (1996b). Solve the problem as a continuous variable optimization problem.

15 Discrete Variable Optimum Design Concepts and Methods

Upon completion of this chapter, you will be able to:

- Formulate mixed continuous-discrete variable optimum design problems
- Use the terminology associated with mixed continuous-discrete variable optimization problems
- Explain concepts associated with various types of mixed continuous-discrete variable optimum design problems and methods
- Determine an appropriate method to solve your mixed continuous-discrete variable optimization problem

Discrete Variable. A variable is called discrete if its value must be assigned from a given set of values.

Integer Variable. A variable that can have only integer values is called an integer variable. Note that the integer variables are just a special class of discrete variables.

Linked Discrete Variable. If assignment of a value to a variable specifies the values for a group of parameters, then it is called a linked discrete variable.

Binary Variable. A discrete variable that can have a value of 0 or 1 is called a binary variable.

In many practical applications, discrete and integer design variables occur naturally in the problem formulation. For example, plate thickness must be selected from the available ones, number of bolts must be an integer, material properties must correspond to the available materials, number of teeth in a gear must be an integer, number of reinforcing bars in a concrete member must be an integer, diameter of reinforcing bars must be selected from the available ones, number of strands in a prestressed member must be an integer, structural members must be selected from commercially available ones, and many more. Types of discrete variables and cost and constraint functions can dictate the method used to solve such problems. For the sake of brevity, we shall refer to these problems as mixed variable (discrete, continuous, integer) optimization problems, or in short MV-OPT. In this chapter, we shall describe various types of MV-OPT problems, and concepts and terminologies associated with their solution. Various methods for solution of different types of problems shall be described. The approach

taken is to stress the basic concepts of the methods and point out their advantages and disadvantages.

Because of the importance of this class of problems for practical applications, considerable interest has been shown in the literature to study and develop appropriate methods for their solution. Material for the present chapter is introductory in nature and describes various solution strategies in the most basic form. The material is derived from several publications of the author and his coworkers, and numerous other references cited there (Arora *et al.*, 1994; Arora and Huang, 1996; Huang and Arora, 1995, 1997a,b; Huang *et al.*, 1997; Arora, 1997, 2002; Kocer and Arora 1996a,b, 1997, 1999, 2002). These references contain numerous examples of various classes of discrete variable optimization problems. Only a few of these examples are covered in this chapter.

15.1 Basic Concepts and Definitions

15.1.1 Definition of Mixed Variable Optimum Design Problem: MV-OPT

The standard design optimization model defined and treated in earlier chapters with equality and inequality constraints can be extended by defining some of the variables as continuous and others as discrete, as follows (MV-OPT):

$$\begin{aligned} & \text{minimize } f(\mathbf{x}) \text{ subject to} \\ & h_i = 0, \quad i = 1 \text{ to } p \\ & g_j \leq 0, \quad j = 1 \text{ to } m \\ & x_i \in D_i, D_i = (d_{i1}, d_{i2}, \dots, d_{iq_i}), \quad i = 1 \text{ to } n_d \\ & x_{il} \leq x_i \leq x_{iu}, \quad i = (n_d + 1) \text{ to } n \end{aligned} \quad (15.1)$$

where f , h_i , and g_j are cost and constraint functions, respectively; x_{il} and x_{iu} are lower and upper bounds for the continuous design variable x_i ; p , m , and n are the number of equality constraints, inequality constraints, and design variables, respectively; n_d is the number of *discrete design variables*; D_i is the set of discrete values for the i th variable; q_i is the number of allowable discrete values; and d_{ik} is the k th possible discrete value for the i th variable. Note that the foregoing problem definition includes *integer variable* as well as *0-1 variable* problems. The formulation in Eq. (15.1) can also be used to solve design problems with linked discrete variables (Arora and Huang 1996; Huang and Arora 1997a). There are many design applications where such linked discrete variables are encountered. We shall describe some of them in a later section.

15.1.2 Classification of Mixed Variable Optimum Design Problems

Depending on the type of design variables, and cost and constraint functions, the mixed continuous-discrete variable problems can be classified into five different categories as discussed later. Depending on the type of the problem, one discrete variable optimization method may be more effective than another to solve the problem. In the following, we assume that the continuous variables in the problem can be treated with an appropriate continuous variable optimization method. Or, if appropriate, a continuous variable is transformed to a discrete variable by defining a grid for it. Thus we focus only on the discrete variables.

MV-OPT 1 Mixed design variables; problem functions are twice continuously differentiable; discrete variables can have nondiscrete values during the solution process (i.e., functions can be evaluated at nondiscrete points). Several solution strategies are available for this class of problem. There are numerous examples of this type of problem; e.g., plate thickness from specified values and member radii from the ones available in the market.

MV-OPT 2 Mixed design variables; problem functions are nondifferentiable; however, discrete variables can have nondiscrete values during the solution process. An example of this class of problems includes design problems where constraints from a design code are imposed. Many times, these constraints are based on experiments and experience, and are not differentiable everywhere in the feasible set. Another example is given in Huang and Arora (1997a,b).

MV-OPT 3 Mixed design variables; problem functions may or may not be differentiable; some of the discrete variables must have only discrete values in the solution process; some of the problem functions can be evaluated only at discrete design variable values during the solution process. Examples of such variables are: number of strands in a prestressed beam or column, number of teeth in a gear, and the number of bolts for a joint. On the other hand, a problem is not classified as MV-OPT 3 if the effects of the nondiscrete design points can be “simulated” somehow. For instance, a coil spring must have an integer number of coils. However, during the solution process, having a noninteger number of coils is acceptable (it may or may not have any physical meaning) as long as function evaluations are possible.

MV-OPT 4 Mixed design variables; problem functions may or may not be differentiable; some of the discrete variables are linked to others; assignment of a value to one variable specifies values for others. This type of a problem covers many practical applications, such as structural design with members selected from a catalog, material selection, and engine-type selection.

MV-OPT 5 Combinatorial problems. These are purely discrete nondifferentiable problems. A classic example of this class of problems is the traveling salesman problem. The total distance traveled to visit a number of cities needs to be minimized. A set of integers (cities) can be arranged in different orders to specify a travel schedule (a design). A particular integer can appear only once in a sequence. Examples of this type of engineering design problems include design of a bolt insertion sequence, welding sequence, and member placement sequence between given set of nodes (Huang *et al.*, 1997).

As will be seen later, some of the discrete variable methods assume that the functions and their derivatives can be evaluated at nondiscrete points. Such methods are not applicable to some of the problem types defined above. Various characteristics of the five problem types are summarized in Table 15-1.

15.1.3 Overview of Solution Concepts

Enumerating on the allowable discrete values for each of the design variables can always solve discrete variable optimization problems. The number of combinations N_c to be evaluated in such a calculation is given as

$$N_c = \prod_{i=1}^{n_d} q_i \quad (15.2)$$

The number of combinations to be analyzed, however, increases very rapidly with an increase in n_d , the number of design variables, and q_i , the number of allowable discrete values for each

TABLE 15-1 Characteristics of Design Variables and Functions for Problem Types

<i>MV-OPT</i>	<i>Variable types</i>	<i>Functions differentiable?</i>	<i>Functions defined at nondiscrete points?</i>	<i>Nondiscrete values allowed for discrete variables?</i>	<i>Are discrete variables linked?</i>
1	Mixed	Yes	Yes	Yes	No
2	Mixed	No	Yes	Yes	No
3	Mixed	Yes/No	No	No	No
4	Mixed	Yes/No	No	No	Yes
5	Discrete	No	No	No	Yes/No

variable. This can lead to an extremely large computational effort to solve the problem. Thus many discrete variable optimization methods try to reduce the search to only a partial list of possible combinations using various strategies and heuristic rules. This is sometimes called *implicit enumeration*. Most of the methods guarantee optimum solution for only a very restricted class of problems (linear or convex). For more general nonlinear problems, however, good usable solutions can be obtained depending on how much computation is allowed. Note that at a discrete optimum point, none of the inequalities may be active unless the discrete point happens to be exactly on the boundary of the feasible set. Also the final solution is affected by how widely separated the allowable discrete values are in the sets D_i in Eq. (15.1).

It is important to note that if the problem is MV-OPT 1 type, then it is useful to solve it first using a continuous variable optimization method. *The optimum cost function value for the continuous solution represents a lower bound for the value corresponding to a discrete solution.* The requirement of discreteness of design variables represents additional constraints on the problem. Therefore, the optimum cost function with discrete design variables will have higher value compared with that for the continuous solution. This way the penalty paid to have a discrete solution can be assessed.

There are two basic classes of methods for MV-OPT: enumerative and stochastic. In the enumerative category full enumeration is a possibility; however partial enumeration is most common based on branch and bound methods. In the stochastic category, the most common ones are simulated annealing and genetic algorithms. Simulated annealing will be discussed later in this chapter, and genetic algorithms will be discussed in Chapter 16.

15.2 Branch and Bound Methods (BBM)

The branch and bound (BBM) method was originally developed for discrete variable linear programming (LP) problems for which a global optimum solution is obtained. It is sometimes called an *implicit enumeration* method because it reduces the full enumeration in a systematic manner. It is one of the earliest and the best-known methods for discrete variable problems and has also been used to solve MV-OPT problems. The concepts of *branching*, *bounding*, and *fathoming* are used to perform the search, as explained later. The following definitions are useful for description of the method, especially when applied to continuous variable problems.

Half-Bandwidth. When r allowable discrete values are taken below and $(r - 1)$ values are taken above a given discrete value for a variable, giving $2r$ allowable values, the parameter r is called the half-bandwidth. It is used to limit the number of allowable

values for a discrete variable, for example, based on the rounded-off continuous solution.

Completion. Assignment of discrete values from the allowable ones to all the variables is called a completion.

Feasible Completion. It is a completion that satisfies all the constraints.

Partial Solution. It is an assignment of discrete values to some but not all the variables for a continuous discrete problem.

Fathoming. A partial solution for a continuous problem or a discrete intermediate solution for a discrete problem (*node of the solution tree*) is said to be fathomed if it is determined that no feasible completion of smaller cost than the one previously known can be determined from the current point. It implies that all possible completions have been *implicitly enumerated* from this node.

15.2.1 Basic BBM

The first use of the branch and bound method for linear problems is attributed to Land and Doig (1960). Dakin (1965) later modified the algorithm that has been subsequently used for many applications. There are two basic implementations of the BBM. In the first one, nondiscrete values for the discrete variables are not allowed (or they are not possible) during the solution process. This implementation is quite straightforward; the concepts of *branching*, *bounding*, and *fathoming* are used directly to obtain the final solution. No subproblems are defined or solved; only the problem functions are evaluated for different combinations of design variables. In the second implementation, nondiscrete values for the design variables are allowed. Forcing a variable to have a discrete value generates a node of the solution tree. This is done by defining additional constraints to force out a discrete value for the variable. The subproblem is solved using either LP or NLP methods depending on the problem type. Example 15.1 demonstrates use of the BBM when only discrete values for the variables are allowed.

EXAMPLE 15.1 BBM with Only Discrete Values Allowed

Solve the following LP problem:

$$\text{minimize } f = -20x_1 - 10x_2 \quad (\text{a})$$

subject to

$$g_1 = -20x_1 - 10x_2 + 75 \leq 0 \quad (\text{b})$$

$$g_2 = 12x_1 + 7x_2 - 55 \leq 0 \quad (\text{c})$$

$$g_3 = 25x_1 + 10x_2 - 90 \leq 0 \quad (\text{d})$$

$$x_1 \in \{0, 1, 2, 3\}, \text{ and } x_2 \in \{0, 1, 2, 3, 4, 5, 6\} \quad (\text{e})$$

Solution. In this implementation of the BBM, variables x_1 and x_2 can have only discrete values from the given four and seven values, respectively. The full enumeration would require evaluation of problem functions for 28 combinations; however, the BBM can find the final solution in fewer evaluations. For the problem, the derivatives of f with respect to x_1 and x_2 are always negative. This information can be used to

advantage in the BBM. One can enumerate the discrete points in the descending order of x_1 and x_2 to ensure that the cost function is always increased when one of the variables is perturbed to the next lower discrete value. The BBM for the problem is illustrated in Fig. 15-1. For each point (called a *node*), the cost and constraint function values are shown. From each node, assigning the next smaller value to each of the variables generates two more nodes. This is called *branching*. At each node, all the problem functions are evaluated again. If there is any constraint violation at a node, further branching is necessary from that node. Once a feasible completion is obtained, the node requires no further branching since no point with a lower cost is possible from there. Such nodes are said to have *fathomed*, i.e., reached their lowest point on the branch and no further branching will produce a solution with lower cost. Nodes 6 and 7 are fathomed this way where the cost function has a value of -80 . For the remaining nodes, this value becomes an upper bound for the cost function. This is called *bounding*. Later any node having a cost function value higher than the current bound is also fathomed. Nodes 9, 10, and 11 are fathomed because the designs are infeasible with the cost function value larger than or equal to the current bound of -80 . Since no further branching is possible, the global solution for the problem is found at Nodes 6 and 7 in 11 function evaluations.

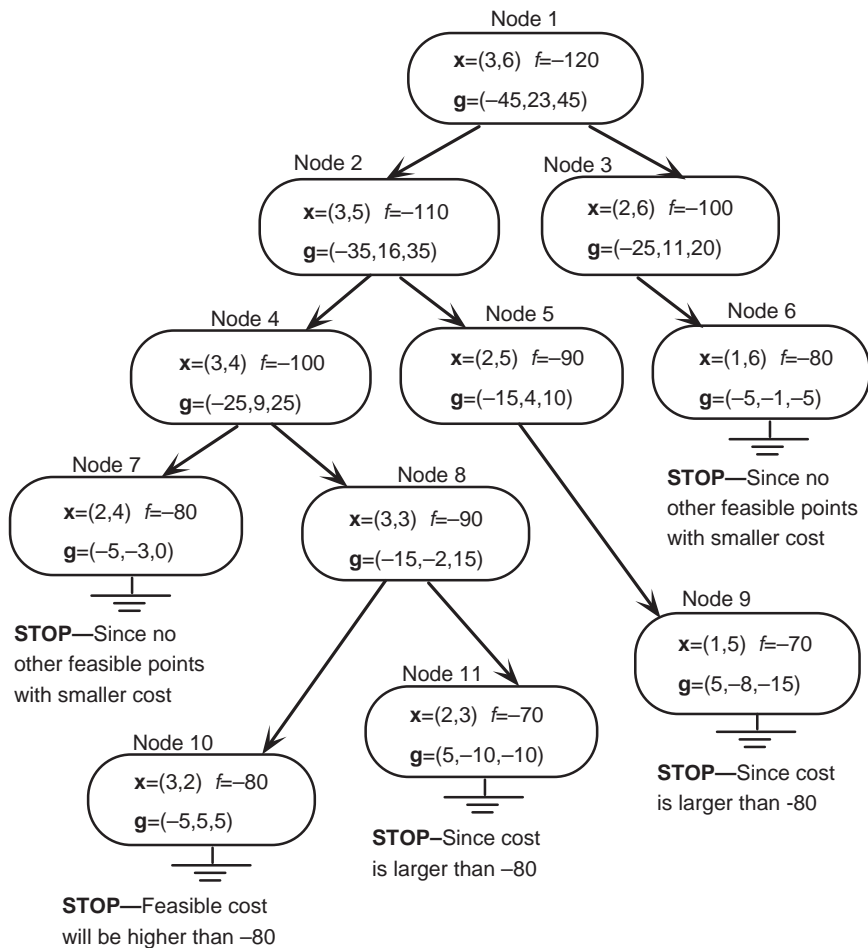


Figure 15-1 Basic branch and bound method without solving continuous subproblems.

15.2.2 BBM with Local Minimization

For optimization problems where the discrete variables can have nondiscrete values during the solution process and all the functions are differentiable, we can take advantage of the local minimization procedures to reduce the number of nodes in the solution tree. In such a BBM procedure, initially an optimum point is obtained by treating all the discrete variables as continuous. If the solution is discrete, an optimum point is obtained and the process is terminated. If one of the variables does not have a discrete value, then its value lies between two discrete values; e.g., $d_{ij} < x_i < d_{ij+1}$. Now two subproblems are defined, one with the constraint $x_i \leq d_{ij}$ and the other with $x_i \geq d_{ij+1}$. This process is also called *branching*, which is slightly different from the one explained in Example 15.1 for purely discrete problems. It basically eliminates some portion of the continuous feasible region that is not feasible for the discrete problem. However, none of the discrete feasible solutions is eliminated. The two subproblems are solved again, and the optimum solutions are stored as nodes of the tree containing optimum values for all the variables, the cost function, and the appropriate bounds on the variables. This process of branching and solving continuous problems is continued until a feasible discrete solution is obtained. Once this has been achieved, the cost function corresponding to the discrete feasible solution becomes an *upper bound* on the cost function for the remaining subproblems (nodes) to be solved later. The solutions that have cost values higher than the upper bound are eliminated from further consideration (i.e., they are fathomed).

The foregoing process of branching and fathoming is repeated from each of the unfathomed nodes. The search for the optimum solution terminates when all the nodes have been fathomed as a result of one of the following reasons: (1) a discrete optimum solution is found, (2) no feasible continuous solution can be found, or (3) a feasible solution is found but the cost function value is higher than the established upper bound. Example 15.2 illustrates use of the BBM where nondiscrete values for the variables are allowed during the solution process.

EXAMPLE 15.2 BBM with Local Minimizations

Re-solve the problem of Example 15.1 treating the variables as continuous during the branching and bounding process.

Solution. Figure 15-2 shows implementation of the BBM where requirements of discreteness and nondifferentiability of the problem functions are relaxed during the solution process. Here one starts with a continuous solution for the problem. From that solution two subproblems are defined by imposing an additional constraint requiring that x_1 not be between 1 and 2. Subproblem 1 imposes the constraint $x_1 \leq 1$ and Subproblem 2, $x_1 \geq 2$. Subproblem 1 is solved using the continuous variable algorithm that gives a discrete value for x_1 but a nondiscrete value for x_2 . Therefore further branching is needed from this node. Subproblem 2 is also solved using the continuous variable algorithm that gives discrete values for the variables with the cost function of -80 . This gives an upper bound for the cost function, and no further branching is needed from this node. Using the solution of Subproblem 1, two subproblems are defined by requiring that x_2 be not between 6 and 7. Subproblem 3 imposes the constraint $x_2 \leq 6$ and Subproblem 4, $x_2 \geq 7$. Subproblem 3 has a discrete solution with $f = -80$, which is the same as the current upper bound. Since the solution is discrete, there is no need to branch further from there by defining more subproblems. Sub-

problem 4 does not lead to a discrete solution with $f = -80$. Since further branching from this node cannot lead to a discrete solution with the cost function value smaller than the current upper bound of -80 , the node is fathomed. Thus, Subproblems 2 and 3 give the two optimum discrete solutions for the problem, as before.

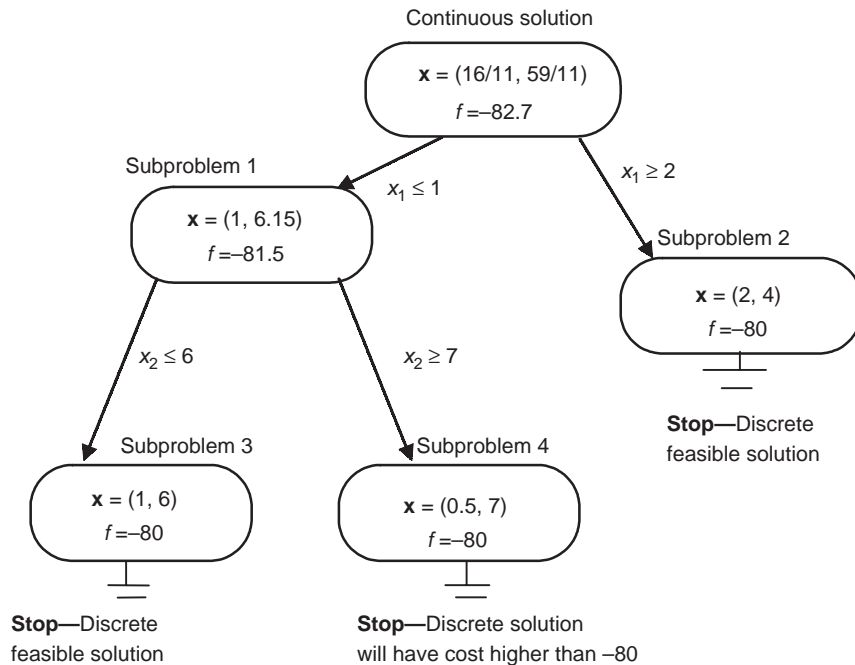


Figure 15-2 Branch and bound method with solution of continuous subproblems.

Since the foregoing problem has only two design variables, it is fairly straightforward to decide how to create various nodes of the solution process. When there are more design variables, node creation and the branching processes are not unique. These aspects are discussed further for nonlinear problems.

15.2.3 BBM for General MV-OPT

In most practical applications for nonlinear discrete problems, the latter version of the BBM has been used most often, where functions are assumed to be differentiable and the design variables can have nondiscrete values during the solution process. Different methods have been used to solve nonlinear optimization subproblems to generate the nodes. The branch and bound method has been used successfully to deal with discrete design variable problems and has proved to be quite robust. However, for problems with a large number of discrete design variables, the number of subproblems (nodes) becomes large. Therefore considerable effort has been spent in investigating strategies to reduce the number of nodes by trying different fathoming and branching rules. For example, the variable that is used for branching to its upper and lower values for the two subproblems is fixed to the assigned value, thus eliminating it from further consideration. This reduces dimensionality of the subproblem that can result in efficiency. As the iterative process progresses, more and more variables are fixed and the size of the optimization problem keeps on decreasing. Many other variations of the

BBM for nonlinear continuous problems have been investigated to improve its efficiency. Since an early establishment of a good upper bound on the cost is important, it may be possible to accomplish this by choosing an appropriate variable for branching. More nodes or subproblems may be fathomed early if a smaller upper bound is established. Several ideas have been investigated in this regard. For example, the distance of a continuous variable from its nearest discrete value, and the cost function value when a variable is assigned a discrete value can be used to decide the variable to be branched.

It is important to note that the BBM is guaranteed to find the global optimum only if the problem is linear or convex. In the case of general nonlinear nonconvex problems, there is no such guarantee. It is possible that a node is fathomed too early and one of its branches actually contains the true global solution.

15.3 Integer Programming

Optimization problems where the variables are required to take on integer values are called integer programming (IP) problems. If some of the variables are continuous, then we get a mixed variable problem. With all functions as linear, an integer linear programming (ILP) problem is obtained, otherwise it is nonlinear. The ILP problem can be converted to a 0-1 programming problem. Linear problems with discrete variables can also be converted to 0-1 programming problems. Several algorithms are available to solve such problems (Sysko *et al.*, 1983; Schrijver, 1986), such as the BBM discussed earlier. Nonlinear discrete problems can also be solved by sequential linearization procedures if the problem functions are continuous and differentiable, as discussed later. In this section, we show how to transform an ILP into a 0-1 programming problem. To do that, let us consider an ILP as follows:

$$\text{minimize } f = \mathbf{c}^T \mathbf{x} \text{ subject to } \mathbf{A} \mathbf{x} \leq \mathbf{b}$$

$$x_i \geq 0 \text{ integer}; i = 1 \text{ to } n_d; \quad x_{i_l} \leq x_i \leq x_{i_u}, i = n_d + 1 \text{ to } n \quad (15.3)$$

Define z_{ij} as the 0-1 variables ($z_{ij} = 0$ or 1 for all i and j). Then the i th integer variable is expressed as

$$x_i = \sum_{j=1}^{q_i} z_{ij} d_{ij}; \quad \sum_j z_{ij} = 1, \quad i = 1 \text{ to } n_d \quad (15.4)$$

where q_i and d_{ij} are defined in Eq. (15.1). Substituting this into the foregoing mixed ILP problem, it is converted to a 0-1 programming problem in terms of z_{ij} , as

$$\begin{aligned} \text{minimize } f = \sum_{i=1}^{n_d} c_i \left[\sum_{j=1}^{q_i} z_{ij} d_{ij} \right] + \sum_{k=n_d+1}^n c_k x_k \quad \text{subject to} \\ \sum_{j=1}^{n_d} a_{ij} \left[\sum_{m=1}^{q_j} z_{jm} d_{jm} \right] + \sum_{k=n_d+1}^n a_{ik} x_k \leq b_i \end{aligned} \quad (15.5)$$

$$\sum_{j=1}^{q_i} z_{ij} = 1, i = 1 \text{ to } n_d; \quad z_{ij} = 0 \text{ or } 1 \text{ for all } i \text{ and } j; \quad x_{i_l} \leq x_i \leq x_{i_u}; i = n_d + 1 \text{ to } n$$

It is important to note that many modern computer programs for linear programming have an option to solve discrete variable LP problems; e.g., LINDO (Schrage, 1991).

15.4 Sequential Linearization Methods

If functions of the problem are differentiable, a reasonable approach to solving the MV-OPT is to linearize the nonlinear problem at each iteration. Then discrete-integer linear programming (LP) methods can be used to solve the linearized subproblem. There are several ways in which the linearized subproblems can be defined and solved. For example, the linearized subproblem can be converted to a 0-1 variable problem. This way the number of variables is increased considerably; however, several methods are available to solve integer linear programming problems. Therefore, MV-OPT can be solved using the sequential LP approach and existing codes. A modification of this approach is to obtain a continuous optimum point first, and then linearize and use integer programming methods. This process can reduce the number of integer LP subproblems to be solved. Restricting the number of discrete values to those in the neighborhood of the continuous solution (a small value for r , the half bandwidth) can also reduce the size of the ILP problem. It is noted here that once a continuous solution has been obtained, then any discrete variable optimization method can be used with a reduced set of discrete values for the variables.

Another possible approach to solve an MV-OPT problem is to optimize for discrete and continuous variables in sequence. The problem is first linearized in terms of the discrete variables but keeping the continuous variables fixed at their current values. The linearized discrete subproblem is solved using a discrete variable optimization method. Then the discrete variables are fixed at their current values, and the continuous subproblem is solved using a nonlinear programming method. The process is repeated a few times to obtain the final solution.

15.5 Simulated Annealing

Simulated annealing (SA) is a stochastic approach that simulates the statistical process of growing crystals using the annealing process to reach its absolute (global) minimum internal energy configuration. If the temperature in the annealing process is not lowered slowly and enough time is not spent at each temperature, the process could get trapped in a local minimum state for the internal energy. The resulting crystal may have many defects or the material may even become glass with no crystalline order. *The simulated annealing method for optimization of systems emulates this process.* Given a long enough time to run, an algorithm based on this concept finds global minima for continuous-discrete-integer variable nonlinear programming problems.

The basic procedure for implementation of this analogy to the annealing process is to generate random points in the neighborhood of the current best point and evaluate the problem functions there. If the cost function (*penalty function for constrained problems*) value is smaller than its current best value, then the point is accepted, and the best function value is updated. If the function value is higher than the best value known thus far, then the point is sometimes accepted and sometimes rejected. Point's acceptance is based on the value of the probability density function of the Boltzman-Gibbs distribution. If this probability density function has a value greater than a random number, then the trial point is accepted as the best solution even if its function value is higher than the known best value. In computing the probability density function, a parameter called the *temperature* is used. For the optimization problem, this temperature can be a target for the optimum value of the cost function. Initially, a larger target value is selected. As the trials progress, the target value (the temperature) is reduced (this is called the *cooling schedule*), and the process is terminated after a large number of trials. The acceptance probability steadily decreases to zero as the temperature is reduced. Thus, in the initial stages, the method sometimes accepts worse

designs, while in the final stages, the worse designs are almost always rejected. This strategy avoids getting trapped at a local minimum point.

It is seen that the SA method requires evaluation of cost and constraint functions only. Continuity and differentiability of functions are not required. Thus the method can be useful for nondifferentiable problems, and problems for which gradients cannot be calculated or are too expensive to calculate. It is also possible to implement the algorithm on parallel computers to speed up the calculations. The deficiencies of the method are the unknown rate for reduction of the target level for the global minimum, and the uncertainty in the total number of trials and the point at which the target level needs to be reduced.

Simulated Annealing Algorithm It is seen that the algorithm is quite simple and easy to program. The following steps illustrate the basic ideas of the algorithm.

- Step 1.* Choose an initial temperature T_0 (expected global minimum for the cost function) and a feasible trial point $\mathbf{x}^{(0)}$. Compute $f(\mathbf{x}^{(0)})$. Select an integer L (e.g., a limit on the number of trials to reach the expected minimum value), and a parameter $r < 1$. Initialize the iteration counter as $K = 0$ and another counter $k = 1$.
- Step 2.* Generate a new point $\mathbf{x}^{(k)}$ randomly in a neighborhood of the current point. If the point is infeasible, generate another random point until feasibility is satisfied (a variation of this step is explained later). Compute $f(\mathbf{x}^{(k)})$ and $\Delta f = f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(0)})$.
- Step 3.* If $\Delta f < 0$, then take $\mathbf{x}^{(k)}$ as the new best point $\mathbf{x}^{(0)}$, set $f(\mathbf{x}^{(0)}) = f(\mathbf{x}^{(k)})$ and go to Step 4. Otherwise, calculate the probability density function:

$$p(\Delta f) = \exp\left(\frac{\Delta f}{T_K}\right) \quad (15.6)$$

Generate a random number z uniformly distributed in $[0,1]$. If $z < p(\Delta f)$, then take $\mathbf{x}^{(k)}$ as the new best point $\mathbf{x}^{(0)}$ and go to Step 4. Otherwise go to Step 2.

- Step 4.* If $k < L$, then set $k = k + 1$ and go to Step 2. If $k > L$ and any of the stopping criteria is satisfied, then stop. Otherwise, go to Step 5.
- Step 5.* Set $K = K + 1$, $k = 1$; set $T_K = rT_{K-1}$; go to Step 2.

The following points are noted for implementation of the algorithm:

1. In Step 2 only one point is generated at a time within a certain neighborhood of the current point. Thus, although SA randomly generates design points without the need for function or gradient information, it is not a pure random search within the entire design space. At the early stage, a new point can be located far away from the current point to speed up the search process and to avoid getting trapped at a local minimum point. Once the temperature gets low, the new point is usually created nearby in order to focus on the local area. This can be controlled by defining a step size procedure.
2. In Step 2, the newly generated point is required to be feasible. If it is not, another point is generated until feasibility is attained. Another method for treating constraints is to use the penalty function approach; i.e., the constrained problem is converted to an unconstrained one, as discussed in Chapter 9. The cost function is replaced by the penalty function in the algorithm. Therefore the feasibility requirements are not imposed explicitly in Step 2.
3. The following stopping criteria are suggested in Step 4: (1) The algorithm stops if change in the best function value is less than some specified value for the last J consecutive iterations. (2) The *program* stops if $I/L < \delta$, where L is a limit on the

number of trials (or number of feasible points generated) within one iteration, and I is the number of trials that satisfy $\Delta f < 0$ (see Step 3). (3) The algorithm stops if K reaches a preset value.

The foregoing ideas from statistical mechanics can also be used to develop methods for global optimization of continuous variable problems. For such problems, simulated annealing may be combined with a local minimization procedure. However, the temperature T is slowly and continuously decreased so that the effect is similar to annealing. Using the probability density function given in Eq. (15.6), a criterion can be used to decide whether to start a local search from a particular point.

15.6 Dynamic Rounding-off Method

A simple approach for MV-OPT 1 type problems is to first obtain an optimum solution using a continuous approach. Then, using heuristics, the variables are rounded off to their nearest available discrete values to obtain a discrete solution. Rounding-off is a simple idea that has been used often, but it can result in infeasible designs for problems having a large number of variables. The main concern of the rounding-off approach is the selection of variables to be increased and the variables to be decreased. The strategy may not converge, especially in case of high nonlinearity and widely separated allowable discrete values. In that case, the discrete minimum point need not be in a neighborhood of the continuous solution.

Dynamic Rounding-off Algorithm The dynamic rounding-off algorithm is a simple extension of the usual rounding-off procedure. The basic idea is to round off variables in a sequence rather than all of them at the same time. After a continuous variable optimum solution is obtained, one or a few variables are selected for discrete assignment. This assignment can be based on the penalty that needs to be paid for the increase in the cost function or the Lagrangian function. These variables are then eliminated from the problem and the continuous variable optimization problem is solved again. This idea is quite simple because an existing optimization program can be used to solve discrete variable problem of type MV-OPT 1. The process can be carried out in an interactive mode, as demonstrated in Chapter 14 for a structural design problem, or it may be implemented manually. Whereas the dynamic rounding-off strategy can be implemented in many different ways, the following algorithm illustrates one simple procedure:

- Step 1.* Assume all the design variables to be continuous and solve the NLP problem.
- Step 2.* If the solution is discrete, stop. Otherwise, continue.
- Step 3.* FOR $k = 1$ to n
 - Calculate the Lagrangian function value for each k with the k th variable perturbed to its discrete neighbors.
 - END FOR
- Step 4.* Choose a design variable that minimizes the Lagrangian in Step 3 and remove that variable from the design variable set. This variable is assigned the selected discrete value. Set $n = n - 1$ and if $n = 1$, stop; otherwise, go to Step 2.

The number of additional continuous problems that needs to be solved by the above method is $(n - 1)$. However, the number of design variables is reduced by one for each subsequent continuous problem. In addition, more variables may be assigned discrete values each time, thus reducing the number of continuous problems to be solved. The dynamic

rounding-off strategy has been used successfully to solve several optimization problems (Section 14.7; Al-Saadoun and Arora, 1989; Huang and Arora, 1997a,b).

15.7 Neighborhood Search Method

When the number of discrete variables is small and each discrete variable has only a few choices, the simplest way to find the solution of a mixed variable problem may be just to explicitly enumerate all the possibilities. With all the discrete variables fixed at their chosen values, the problem is then optimized for the continuous variables. This approach has some advantages over the BBM: it can be implemented easily with an existing optimization program, the problem to be solved is smaller, and the gradient information with respect to the discrete variables is not needed. However, the approach is far less efficient than an implicit enumeration method, such as the BBM, as the number of discrete variables and size of the discrete set of values become large.

When the number of discrete variables is large and the number of discrete values for each variable is large, then a simple extension of the above approach is to solve the optimization problem first by treating all the variables as continuous. Based on that solution, a reduced set of allowable discrete values for each variable is then selected. Now the neighborhood search approach is used to solve the MV-OPT 1 problem. A drawback is that the search for a discrete solution is restricted to only a small neighborhood of the continuous solution.

15.8 Methods for Linked Discrete Variables

Linked discrete variables occur in many applications. For example, in the design of a coil spring problem formulated in Chapter 2, one may have choice of three materials as shown in Table 15-2. Once a material type is specified, all the properties associated with it must be selected and used in all calculations. The optimum design problem is to determine the material type and other variables to optimize an objective function and satisfy all the constraints. The problem has been solved in Huang and Arora (1997a,b).

Another practical example where linked discrete variables are encountered is the optimum design of framed structural systems. Here the structural members must be selected from the ones available in manufacturer's catalog. Table 15-3 shows some of the standard sections available in the catalog. The optimum design problem is to find the best possible sections for members of a structural frame to minimize a cost function and satisfy all the performance constraints. The section number, section area, moment of inertia, or any other section property can be designated as a linked discrete design variable for the frame member. Once a value for such a discrete variable is specified from the table, each of its linked variables (properties) must also be assigned the unique value and used in the optimization process. These properties affect values of the cost and constraint functions for the problem. A certain value for a particular property can only be used when appropriate values for other properties

TABLE 15-2 Material Data for Spring Design Problem

	G , lb/in ²	ρ , lb-s ² /in ⁴	τ_a , lb/in ²	U_p
Material type 1	11.5×10^6	7.38342×10^{-4}	80,000	1.0
Material type 2	12.6×10^6	8.51211×10^{-4}	86,000	1.1
Material type 3	13.7×10^6	9.71362×10^{-4}	87,000	1.5

G = shear modulus, ρ = mass density, τ_a = shear stress, U_p = relative unit price.

TABLE 15-3 Some Wide Flange Standard Sections

Section	A	d	t_w	b	t_f	I_x	S_x	r_x	I_y	S_y	r_y
W36 × 300	88.30	36.74	0.945	16.655	1.680	20300	1110	15.20	1300	156	3.830
W36 × 280	82.40	36.52	0.885	16.595	1.570	18900	1030	15.10	1200	144	3.810
W36 × 260	76.50	36.26	0.840	16.550	1.440	17300	953	15.00	1090	132	3.780
W36 × 245	72.10	36.08	0.800	16.510	1.350	16100	895	15.00	1010	123	3.750
W36 × 230	67.60	35.90	0.760	16.470	1.260	15000	837	14.90	940	114	3.730
W36 × 210	61.80	36.69	0.830	12.180	1.360	13200	719	14.60	411	67.5	2.580
W36 × 194	57.00	36.49	0.765	12.115	1.260	12100	664	14.60	375	61.9	2.560

A : Cross-sectional area, in²

d : Depth, in

t_w : Web thickness, in

b : Flange width, in

t_f : Flange thickness, in

I_x : Moment of inertia about the x - x axis, in⁴

S_x : Elastic section modulus about the x - x axis, in³

r_x : Radius of gyration with respect to the x - x axis, in

I_y : Moment of inertia about the y - y axis, in⁴

S_y : Elastic section modulus about the y - y axis, in³

r_y : Radius of gyration with respect to the y - y axis, in

are also assigned. Relationships among such variables and their linked properties cannot be expressed analytically, and so a gradient-based optimization method may be applicable only after some approximations. It is not possible to use one of the properties as the only continuous design variable because other section properties cannot be calculated using just that property. Also, if each property were treated as an independent design variable, the final solution would generally be unacceptable since the variables would have values that cannot co-exist in the table. Solutions for such problems are presented in Huang and Arora (1997a,b).

It is seen that problems with linked variables are discrete and the problem functions are not differentiable with respect to them. Therefore they must be treated by a discrete variable optimization algorithm that does not require gradients of functions. There are two algorithms for such problems: simulated annealing and genetic algorithms. Simulated annealing has been discussed earlier and genetic algorithms are presented in Chapter 16.

It is noted that for each class of problems having linked discrete variables, it is possible to develop strategies to treat the problem more efficiently by exploiting the structure of the problem and knowledge of the problem functions. Two or more algorithms may be combined to develop strategies that are more effective than the use of a purely discrete algorithm. For the structural design problem, several such strategies have been developed (Arora, 2002).

15.9 Selection of a Method

Selection of a method to solve a particular mixed variable optimization problem depends on the nature of the problem functions. Features of the methods and their suitability for various types of MV-OPT problems are summarized in Table 15-4. It is seen that branch and bound, simulated annealing, and genetic algorithms (discussed in Chapter 16) are the most general methods. They can be used to solve all the problem types. However, these are also the most expensive ones in terms of computational effort. If the problem functions are differentiable and discrete variables can be assigned nondiscrete values during the iterative solution process, then there are numerous strategies for their solution that are more efficient than the three methods just discussed. Most of these involve a combination of two or more algorithms.

Huang and Arora (1997a,b) have evaluated the discrete variable optimization methods presented in this chapter using 15 different types of test problems. Applications involving

TABLE 15-4 Characteristics of Discrete Variable Optimization Methods

	<i>MV-OPT problem type solved</i>	<i>Can find feasible discrete solution?</i>	<i>Can find global minimum for convex prob.?</i>	<i>Need gradients?</i>
Branch and bound	1 – 5	Yes	Yes	No/Yes
Simulated annealing	1 – 5	Yes	Yes	No
Genetic algorithm	1 – 5	Yes	Yes	No
Sequential linearization	1	Yes	Yes	Yes
Dynamic round-off	1	Yes	No Guar.	Yes
Neighborhood search	1	Yes	Yes	Yes

linked discrete variables are described in Huang and Arora (1997), Arora and Huang (1996), and Arora (2002). Applications of discrete variable optimization methods to electric transmission line structures are described in Kocer and Arora (1996, 1997, 1999, 2002). Discrete variable optimum solutions for the plate girder design problem formulated and solved in Section 10.6 are described and discussed in Arora and coworkers (1997).

Exercises for Chapter 15*

- 15.1 Solve Example 15.1 with the available discrete values for the variables as $x_1 \in \{0,1,2,3\}$, and $x_2 \in \{0,1,2,3,4,5,6\}$. Assume that the functions of the problem are not differentiable.
- 15.2 Solve Example 15.1 with the available discrete values for the variables as $x_1 \in \{0,1,2,3\}$, and $x_2 \in \{0,1,2,3,4,5,6\}$. Assuming that the functions of the problem are differentiable, use a continuous variable optimization procedure to solve for discrete variables.
- 15.3 Formulate and solve Exercise 3.34 using the outside diameter d_0 and the inside diameter d_i as design variables. The outside diameter and thickness must be selected from the following available sets:

$$d_0 \in \{0.020, 0.022, 0.024, \dots, 0.48, 0.50\} \text{ m}; \quad t \in \{5, 7, 9, \dots, 23, 25\} \text{ mm}$$

Check your solution using the graphical method of Chapter 3. Compare continuous and discrete solutions.

- 15.4 Consider the minimum mass tubular column problem formulated in Section 2.7. Find the optimum solution for the problem using the following data: $P = 100$ kN, length, $l = 5$ m, Young's modulus, $E = 210$ GPa, allowable stress, $\sigma_a = 250$ MPa, mass density, $\rho = 7850$ kg/m³, $R \leq 0.4$ m, $t \leq 0.05$ m, and $R, t \geq 0$. The design variables must be selected from the following sets:

$$R \in \{0.01, 0.012, 0.014, \dots, 0.38, 0.40\} \text{ m}; \quad t \in \{4, 6, 8, \dots, 48, 50\} \text{ mm}$$

Check your solution using the graphical method of Chapter 3. Compare continuous and discrete solutions.

- 15.5 Consider the plate girder design problem described and formulated in Section 10.6. The design variables for the problem must be selected from the following sets

$$h, b \in \{0.30, 0.31, 0.32, \dots, 2.49, 2.50\} \text{ m}; \quad t_w, t_f \in \{10, 12, 14, \dots, 98, 100\} \text{ mm}$$

Assume that the functions of the problem are differentiable and a continuous variable optimization program can be used to solve subproblems, if needed. Solve the discrete variable optimization problem. Compare the continuous and discrete solutions.

- 15.6 Consider the plate girder design problem described and formulated in Section 10.6. The design variables for the problem must be selected from the following sets

$$h, b \in \{0.30, 0.31, 0.32, \dots, 2.49, 2.50\} \text{ m}; \quad t_w, t_f \in \{10, 12, 14, \dots, 98, 100\} \text{ mm}$$

Assume functions of the problem to be nondifferentiable. Solve the discrete variable optimization problem. Compare the continuous and discrete solutions.

- 15.7 Consider the plate girder design problem described and formulated in Section 10.6. The design variables for the problem must be selected from the following sets

$$h, b \in \{0.30, 0.31, 0.32, \dots, 2.48, 2.50\} \text{ m}; \quad t_w, t_f \in \{10, 14, 16, \dots, 96, 100\} \text{ mm}$$

Assume that the functions of the problem are differentiable and a continuous variable optimization program can be used to solve subproblems, if needed. Solve the discrete variable optimization problem. Compare the continuous and discrete solutions.

- 15.8 Consider the plate girder design problem described and formulated in Section 10.6. The design variables for the problem must be selected from the following sets

$$h, b \in \{0.30, 0.32, 0.34, \dots, 2.48, 2.50\} \text{ m}; \quad t_w, t_f \in \{10, 14, 16, \dots, 96, 100\} \text{ mm}$$

Assume functions of the problem to be nondifferentiable. Solve the discrete variable optimization problem. Compare the continuous and discrete solutions.

- 15.9 Solve the problems of Exercises 15.3 and 15.5. Compare the two solutions, commenting on the effect of the size of the discreteness of variables on the optimum solution. Also, compare the continuous and discrete solutions.
- 15.10 Consider the spring design problem formulated in Section 2.9 and solved in Section 13.5. Assume that the wire diameters are available in increments of 0.01 in, the coils can be fabricated in increments of $\frac{1}{16}$ th of an inch, and the number of coils must be an integer. Assume functions of the problem to be differentiable. Compare the continuous and discrete solutions.
- 15.11 Consider the spring design problem formulated in Section 2.9 and solved in Section 13.5. Assume that the wire diameters are available in increments of 0.01 in, the coils can be fabricated in increments of $\frac{1}{16}$ th of an inch, and the number of coils must be an integer. Assume the functions of the problem to be nondifferentiable. Compare the continuous and discrete solutions.
- 15.12 Consider the spring design problem formulated in Section 2.9 and solved in Section 13.5. Assume that the wire diameters are available in increments of 0.015 in, the

coils can be fabricated in increments of $\frac{1}{8}$ th of an inch, and the number of coils must be an integer. Assume functions of the problem to be differentiable. Compare the continuous and discrete solutions.

- 15.13 Consider the spring design problem formulated in Section 2.9 and solved in Section 13.5. Assume that the wire diameters are available in increments of 0.015 in, the coils can be fabricated in increments of $\frac{1}{8}$ th of an inch, and the number of coils must be an integer. Assume the functions of the problem to be nondifferentiable. Compare the continuous and discrete solutions.
- 15.14 Solve problems of Exercises 15.8 and 15.10. Compare the two solutions, commenting on the effect of the size of the discreteness of variables on the optimum solution. Also, compare the continuous and discrete solutions.
- 15.15 Formulate the problem of optimum design of prestressed concrete transmission poles described in Kocer and Arora (1996a). Use a mixed variable optimization procedure to solve the problem. Compare the solution to that given in the reference.
- 15.16 Formulate the problem of optimum design of steel transmission poles described in Kocer and Arora (1996b). Solve the problem as a continuous variable optimization problem.
- 15.17 Formulate the problem of optimum design of steel transmission poles described in Kocer and Arora (1996b). Assume that the diameters can vary in increments of 0.5 in and the thicknesses can vary in increments of 0.05 in. Solve the problem as a discrete variable optimization problem.
- 15.18 Formulate the problem of optimum design of steel transmission poles using standard sections described in Kocer and Arora (1997). Compare your solution to the solution given there.
- 15.19 Solve the following mixed variable optimization problem (Hock and Schittkowski, 1981):

minimize

$$f = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

subject to

$$g_1 = 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 127$$

$$g_2 = 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 282$$

$$g_3 = 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 196$$

$$g_4 = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

The first three design variables must be selected from the following sets

$$x_1 \in \{1, 2, 3, 4, 5\}; \quad x_2, x_3 \in \{1, 2, 3, 4, 5\}$$

15.20 Formulate and solve the three-bar truss of Exercise 3.50 as a discrete variable problem where the cross-sectional areas must be selected from the following discrete set:

$$A_i \in \{50, 100, 150, \dots, 4950, 5000\} \text{ mm}^2$$

Check your solution using the graphical method of Chapter 3. Compare continuous and discrete solutions.

16 Genetic Algorithms for Optimum Design

Upon completion of this chapter, you will be able to:

- Explain basic concepts and terminology associated with genetic algorithms
- Explain basic steps of a genetic algorithm
- Use a software based on genetic algorithm to solve your optimum design problem

Genetic algorithms (GA) belong to the class of stochastic search optimization methods, such as simulated annealing method described in Chapter 15. As you get to know basics of the algorithms, you will see that decisions made in most computational steps of the algorithms are based on random number generation. The algorithms use only the function values in the search process to make progress toward a solution without regard to how the functions are evaluated. Continuity or differentiability of the problem functions is neither required nor used in calculations of the algorithms. Therefore, the algorithms are very general and can be applied to all kinds of problems—discrete, continuous, and nondifferentiable. In addition, the methods determine global optimum solutions as opposed to the local solutions determined by a continuous variable optimization algorithm. The methods are easy to use and program since they do not require use of gradients of cost or constraint functions. Drawbacks of the algorithms are that (1) they require a large amount of calculation for even reasonable size problems, or for problems where evaluation of functions itself requires massive calculation, and (2) there is no absolute guarantee that a global solution has been obtained. The first drawback can be overcome to some extent by the use of massively parallel computers. The second drawback can be overcome to some extent by executing the algorithm several times and allowing it to run longer.

In the remaining sections of this chapter, concepts and terminology associated with genetic algorithms are defined and explained. Fundamentals of the algorithm are presented and explained. Although the algorithm can be used for continuous problems, our focus will be on discrete variable optimization problems. Various steps of a genetic algorithm are described that can be implemented in different ways. Most of the material for this chapter is derived from the work of the author and his coworkers and is introductory in nature (Arora *et al.*, 1994; Huang and Arora, 1997; Huang *et al.*, 1997; Arora, 2002). Numerous other good ref-

erences on the subject are available (e.g., Holland, 1975; Goldberg, 1989; Mitchell, 1996; Gen and Cheng, 1997; Coello-Coello, 2002; Osyczka, 2002; Pezeshk and Camp, 2002).

16.1 Basic Concepts and Definitions

Genetic algorithms loosely parallel biological evolution and are based on Darwin's theory of natural selection. The specific mechanics of the algorithm use the language of microbiology, and its implementation mimics genetic operations. We shall explain this in subsequent paragraphs and sections. The *basic idea of the approach* is to start with a set of designs, randomly generated using the allowable values for each design variable. Each design is also assigned a fitness value, usually using the cost function for unconstrained problems or the penalty function for constrained problems. From the current set of designs, a subset is selected randomly with a bias allocated to more fit members of the set. Random processes are used to generate new designs using the selected subset of designs. The size of the set of designs is kept fixed. Since more fit members of the set are used to create new designs, the successive sets of designs have a higher probability of having designs with better fitness values. The process is continued until a stopping criterion is met. In the following paragraphs, some details of implementation of these basic steps are presented and explained. First, we shall define and explain various terms associated with the algorithm.

Population The set of design points at the current iteration is called a population. It represents a group of designs as potential solution points. N_p = number of designs in a population; also called the population size.

Generation An iteration of the genetic algorithm is called a generation. A generation has a population of size N_p that is manipulated in a genetic algorithm.

Chromosome This term is used to represent a design point. Thus a chromosome represents a design of the system, whether feasible or infeasible. It contains values for all the design variables of the system.

Gene This term is used for a scalar component of the design vector; i.e., it represents the value of a particular design variable.

Design Representation A method is needed to represent design variable values in the allowable sets and to represent design points so that they can be used and manipulated in the algorithm. This is called a *schema*, and it needs to be encoded; i.e., defined. Although binary encoding is the most common approach, real-number coding and integer encoding are also possible. Binary encoding implies a string of zeros and ones. Binary strings are also useful because it is easier to explain the operations of the genetic algorithm with them. A binary string of 0's and 1's can represent a design variable. Also, an L-digit string with a 0 or 1 for each digit, where L is the total number of binary digits, can be used to specify a design point. Elements of a binary string are called *bits*; a bit can have a value of 0 or 1. We shall use the term "*V-string*" for a binary string that represents the value of a variable; i.e., component of a design vector (a gene). Also, we shall use the term "*D-string*" for a binary string that represents a design of the system; i.e., a particular combination of n V-strings, where n is the number of design variables. This is also called a *genetic string* (or a chromosome).

An m -digit binary string has 2^m possible 0–1 combinations implying that 2^m discrete values can be represented. The following method can be used to transform a V-string consisting of a combination of m 0's and 1's to its corresponding discrete value of a variable having

N_c allowable discrete values: let m be the smallest integer satisfying $2^m > N_c$; calculate the integer j :

$$j = \sum_{i=1}^m ICH(i)2^{(i-1)} + 1 \quad (16.1)$$

where $ICH(i)$ is the value of the i th digit (either 0 or 1). Thus the j th allowable discrete value corresponds to this 0–1 combination; i.e., the j th discrete value corresponds to this V-string. Note that when $j > N_c$ in Eq. (16.1), the following procedure can be used to adjust j such that $j \leq N_c$:

$$j = \text{INT}\left(\frac{N_c}{2^m - N_c}\right)(j - N_c) \quad (16.2)$$

where $\text{INT}(x)$ is the integer part of x . As an example, consider a problem with three design variables each having $N_c = 10$ possible discrete values. Thus we shall need a 4-digit binary string to represent discrete values for each design variable; i.e., $m = 4$ implying that 16 possible discrete values can be represented. Let a design point $\mathbf{x} = (x_1, x_2, x_3)$ be encoded as the following D-string (genetic string):

$$\left[\begin{array}{ccc} x_1 & x_2 & x_3 \\ |0110| & |1111| & |1101| \end{array} \right] \quad (16.3)$$

Using Eq. (16.1), the j values for the three V-strings are calculated as 7, 16, and 14. Since the last two numbers are larger than $N_c = 10$, they are adjusted by using Eq. (16.2) as 6 and 4, respectively. Thus the foregoing D-string (genetic string) represents a design point where the seventh, sixth, and fourth allowable discrete values are assigned to x_1 , x_2 , and x_3 , respectively.

Initial Generation/Starting Design Set With a method to represent a design point defined, the first population consisting of N_p designs needs to be created. This means that N_p D-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. Several methods can be used for this purpose. The following procedure shows a way to produce a 32-digit D-string:

1. Generate random numbers between 0 and 1 as “0.3468 0254 7932 7612 and 0.6757 2163 5862 3845”.
2. Create a string by combining the two numbers as “3468 0254 7932 7612 6757 2163 5862 3845”.
3. The 32 digits of the above string are converted to 0’s and 1’s by using a rule in which “0” is used for any value between 0 and 4 and “1” for any value between 5 and 9, as “0011 0010 1100 1100 1111 0010 1110 0101”.

Fitness Function The fitness function defines the relative importance of a design. A higher fitness value implies a better design. While the fitness function may be defined in several different ways, it can be defined using the cost function value as follows:

$$F_i = (1 + \varepsilon)f_{\max} - f_i, \quad (16.4)$$

where f_i is the cost function (penalty function value for a constrained problems) for the i th design, f_{\max} is the largest recorded cost (penalty) function value, and ε is a small value (e.g., 2×10^{-7}) to prevent numerical difficulties when F_i becomes 0.

16.2 Fundamentals of Genetic Algorithms

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. The process is continued until a stopping criterion is satisfied or the number of iterations exceeds a specified limit. Three genetic operators are used to accomplish this task: reproduction, crossover, and mutation. *Reproduction* is an operator where an old design (D-string) is copied into the new population according to the design's fitness. There are many different strategies to implement this reproduction operator. This is also called the *selection process*. The *crossover* operator corresponds to allowing selected members of the new population to exchange characteristics of their designs among themselves. Crossover entails selection of starting and ending positions on a pair of randomly selected strings (called *mating strings*), and simply exchanging the string of 0's and 1's between these positions. *Mutation* is the third step that safeguards the process from a complete premature loss of valuable genetic material during reproduction and crossover. In terms of a binary string, this step corresponds to 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.

The foregoing three steps are repeated for successive generations of the population until no further improvement in fitness is attainable. The member in this generation with the highest level of fitness is taken as the optimum design. Some details of the GA algorithm implemented by Huang and Arora (1997a) are described in the sequel.

Reproduction Procedure Reproduction is a process of selecting a set of designs (D-strings) from the current population and carrying them into the next generation. The *selection process* is biased toward more fit members of the current design set (population). Using the fitness value F_i for each design in the set, its probability of selection is calculated as

$$P_i = \frac{F_i}{Q}; \quad Q = \sum_{j=1}^{N_p} F_j \quad (16.5)$$

It is seen that the members with higher fitness value have larger probability of selection. To explain the process of selection, let us consider a roulette wheel with a handle shown in Fig. 16-1. The wheel has N_p segments to cover the entire population, with the size of the i th segment proportional to the probability P_i . Now a random number w is generated between 0 and 1. The wheel is then rotated clockwise, with the rotation proportional to the random number w . After spinning the wheel, the member pointed to by the arrow at the starting location is selected for inclusion in the next generation. In the example shown in Fig. 16-1, member 2 is carried into the next generation. Since the segments on the wheel are sized according to the probabilities P_i , the selection process is biased toward the more fit members of the current population. Note that a member copied to the mating pool remains in the current population for further selection. Thus, the new population may contain identical members and may not contain some of the members found in the current population. This way, the average fitness of the new population is increased.

Crossover Once a new set of designs is determined, *crossover* is conducted as a means to introduce variation into a population. Crossover is the process of combining or mixing two

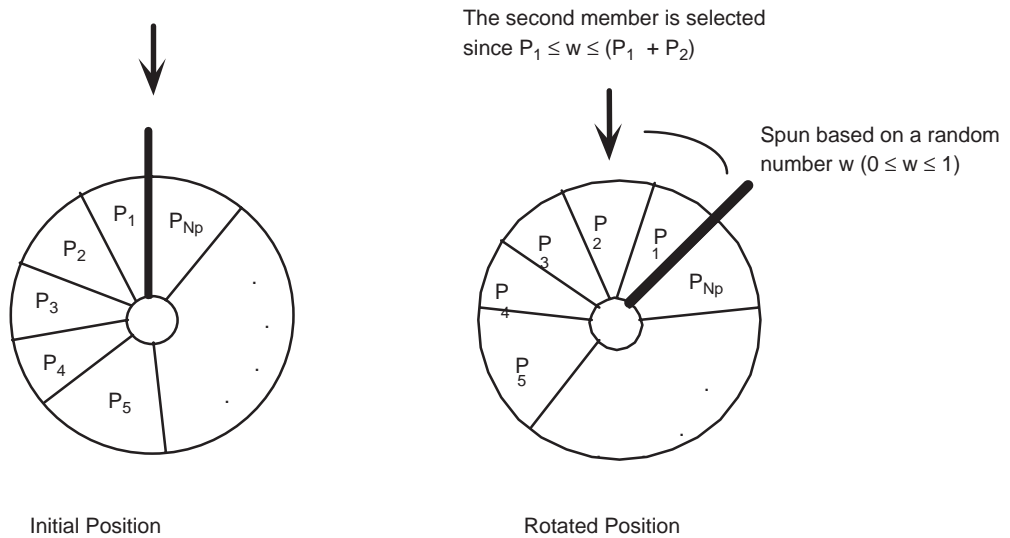


FIGURE 16-1 Roulette wheel process for selection of designs for new generation (reproduction).

$$\mathbf{x}^1 = 101110|1001 \qquad \mathbf{x}^2 = 010100|1011$$

(A) Designs selected for crossover (parent chromosomes)

$$\mathbf{x}^1 = 101110|1011 \qquad \mathbf{x}^2 = 010100|1001$$

(B) New designs (children) after crossover

FIGURE 16-2 Crossover operation with one-cut point.

different designs (chromosomes) in the population. Although there are many methods for performing crossover, the most common ones are the *one-cut-point* and *two-cut-point methods*. A cut point is a position on the D-string (genetic string). In the one-cut method a position on the string is randomly selected that marks the point at which two parent designs (chromosomes) split. The resulting four halves then are exchanged to produce new designs (children). The process is illustrated in Fig. 16-2 where the cut point is determined as 4 digits from the right end. The new designs produced \mathbf{x}^1 and \mathbf{x}^2 and replace the old designs (parents). Similarly, the two-cut-point method is illustrated in Fig. 16-3. Selecting how many or what percentage of chromosomes crossover and at what points the crossover operation occurs are part of the heuristic nature of genetic algorithms. There are many different approaches, and most are based on random selections.

Mutation Mutation is the next operation on the members of the new design set (population). The idea of mutation is to safeguard the process from a complete premature loss of valuable genetic material during reproduction and crossover steps. In terms of a genetic string, this step corresponds to selecting a few members of the population, determining a location on each string randomly, and switching 0 to 1 or vice versa. The number of members

$$\mathbf{x}^1 = 101|1101|001$$

$$\mathbf{x}^2 = 010|1001|011$$

(A) Designs selected for crossover (parent chromosomes)

$$\mathbf{x}' = 101|1001|001$$

$$\mathbf{x}'' = 010|1101|011$$

(B) New designs (children) after crossover

FIGURE 16-3 Crossover operation with two-cut point.

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 as “10 1110 1001” and the location #7 from the right end on its D-string. The mutation operation involves replacing the current value of 1 at the seventh location with 0 as “10 1010 1001”.

Amount of Crossover and Mutation For each generation (iteration), three operators—reproduction or selection, crossover, and mutation—are performed. While the number of the reproduction operations is always equal to the size of the population, the amount of crossover and mutation can be adjusted to fine-tune the performance of the algorithm. To show the type of operations needed to implement the mutation and crossover at each generation, we present a possible procedure as follows.

1. Set I_{\max} as an integer that controls the amount of crossover. Calculate I_m , which controls the amount of mutation as $I_m = \text{INT}(P_m N_p)$, where P_m represents a fraction of the population that is selected for mutation, and N_p is the size of the population. Too many crossovers can result in a poorer performance of the algorithm since it may produce designs that are far away from the mating designs. Therefore, I_{\max} should be set to a small number. The mutation, however, changes designs in the neighborhood of the current design; therefore a larger amount of mutation may be allowed. Note also that the population size N_p 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.
2. Let f_K^* denote the best cost (or penalty) function value for the population at the K th iteration. If the improvement in f_K^* is less than some small positive number ϵ' for the last two consecutive iterations, then I_{\max} is doubled temporarily. This “doubling” strategy continues at the subsequent iterations and returns to the original value as soon as f_K^* is reduced. The concept behind this is that we do not want too much crossover or mutation to ruin the good designs in D-strings as long as they keep producing better offspring. On the other hand, we need more crossover and mutation to trigger changes when progress stops.
3. If the improvement in f_K^* is less than ϵ' for the last I_g consecutive iterations, then P_m is doubled.

4. The crossover and mutation may be performed as follows:

```
FOR  $i = 1, I_{\max}$ 
  Generate a random number  $z$  uniformly distributed in  $[0, 1]$ 
  If  $z > 0.5$ , perform crossover.
  If  $z \leq 0.5$ , skip crossover.
  FOR  $j = 1, I_m$ 
    Generate a random number  $z$  uniformly distributed in  $[0, 1]$ 
    If  $z > 0.5$ , perform mutation.
    If  $z \leq 0.5$ , skip to next  $j$ .
  ENDFOR
ENDFOR
```

Leader of the Population At each generation, the member having the lowest cost function value among all the designs is defined as the “leader” of the population. If several members have the same lowest cost, only one of them is chosen as the leader. The leader is replaced if another member with lower cost appears. This way, it is safeguarded from extinction (as a result of reproduction, crossover, or mutation). In addition, the leader is guaranteed a higher probability of selection for reproduction. One benefit of using a leader is that the best cost (penalty) function value of the population can never increase from one iteration to another, and some of the best design variable values (V-strings or genes) can always survive.

Stopping Criteria If the improvement for the best cost (penalty) function value is less than ϵ for the last I consecutive iterations, or if the number of iterations exceeds a specified value, then the algorithm terminates.

Genetic Algorithm Based on the ideas presented here, a sample genetic algorithm is stated. N_p : population size

- Step 1.* Define a schema to represent different design points. Randomly generate N_p genetic strings (members of the population) according to the schema, where N_p is the population size. Or use the seed designs to generate the initial population. For *constrained problems*, only the feasible strings are accepted when the penalty function approach is not used. Set iteration counter $K = 0$. Define a fitness function for the problem, as in Eq. (16.4).
- Step 2.* Calculate the fitness values for all the designs in the population. Set $K = K + 1$, and the counter for the number of crossovers $I_c = 1$.
- Step 3. Reproduction:* Select designs from the current population according to the roulette wheel selection process described earlier for the mating pool (next generation) from which members for crossover and mutation are selected.
- Step 4. Crossover:* Select two designs from the mating pool. Randomly choose two sites on the genetic strings and swap strings of 0's and 1's between the two chosen sites. Set $I_c = I_c + 1$.
- Step 5. Mutation:* Choose a fraction (P_m) of the members from the mating pool and switch a 0 to 1 or vice versa at a randomly selected site on each chosen string. If, for the past I_g consecutive generations, the member with the lowest cost remains the same, the mutation fraction P_m is doubled. I_g is an integer defined by the user.
- Step 6.* If the member with the lowest cost remains the same for the past two consecutive generations, then increase I_{\max} . If $I_c < I_{\max}$, go to Step 4. Otherwise, continue.
- Step 7. Stopping criterion:* If after the mutation fraction P_m is doubled, the best value of the fitness is not updated for the past I_g consecutive generations, then stop. Otherwise, go to Step 2.

Immigration It may be useful to introduce completely new designs into the population in an effort to increase diversity. This is called immigration, which may be done at a few iterations during the solution process when progress toward the solution point is slow.

Multiple Runs for a Problem It is seen that the genetic algorithms make decisions at several places based on random number generation. Therefore, when the same problem is run at different times, it may give different final designs. It is suggested that the problem be run a few times to ensure that the best possible solution has been obtained.

16.3 Genetic Algorithm for Sequencing-Type Problems

There are many applications in engineering where the sequence of operations needs to be determined. To introduce the type of problems being treated, let us consider the design of a metal plate that is to have 10 bolts at locations shown in Fig. 16-4. Bolts are to be inserted into the predrilled holes by a computer-controlled robot arm. The objective is to minimize the movement of the robot arm while it passes over and inserts a bolt into each hole. This class of problems is generally known as *traveling salesman problem*, which is defined as follows: given a list of N cities and a means to calculate the traveling distance between any two cities, one must plan the salesman route that passes through each city once (with option of returning to the starting point) while minimizing the total distance. For such problems, a feasible design is a string of numbers (a sequence of the cities to be visited) that does not have repeated numbers (e.g., “1 3 4 2” is feasible and “3 1 3 4” is not). Typical operators used in genetic algorithms, such as crossover and mutation, are not applicable to these types of problems since they usually create infeasible designs with repeated numbers. Therefore, other operators need to be used to solve such problems. We shall describe some such operators in the following paragraphs.

Permutation Type 1 Let n_1 be a fraction for selection of the mating pool members for carrying out the Type 1 permutation. Choose Nn_1 members from the mating pool at random, and reverse the sequence between two randomly selected sites on each chosen string. For example, a chosen member with a string of “345216” and two randomly selected sites of “4” and “1”, is changed to “312546”.

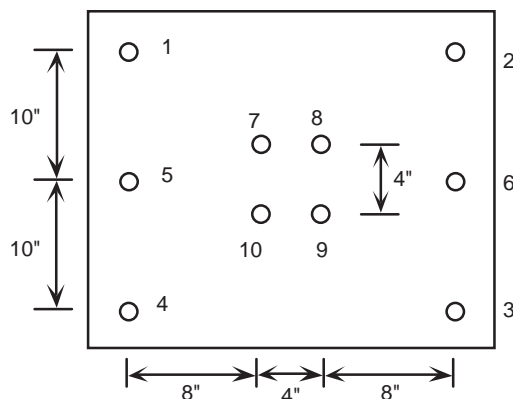


FIGURE 16-4 Bolt insertion sequence determination at 10 locations (Huang *et al.*, 1997).

Permutation Type 2 Let n_2 be a fraction for selection of the mating pool members for carrying out the Type 2 permutation. Choose Nn_2 members from the mating pool at random, and exchange the numbers of two randomly selected sites on each chosen string. For example, a chosen member with a string of “345216” and two randomly selected sites of “4” and “1”, is changed to “315246”.

Permutation Type 3 Let n_3 be a fraction for selection of the mating pool members for carrying out the Type 3 permutation. Choose Nn_3 members from the mating pool at random, and exchange the numbers of one randomly selected site and the site next to it on each chosen string. For example, a chosen member with a string of “345216” and a randomly selected site of “4”, is changed to “354216”.

Relocation Let n_r be a fraction for selection of the mating pool members for carrying out relocation. Choose Nn_r members from the mating pool at random, remove the number of a randomly selected site, and insert it in front of another randomly selected site on each chosen string. For example, a chosen member with a string of “345216” and two randomly selected sites of “4” and “1”, is changed to “352416”.

A computer program based on the previously mentioned operators is developed and used to solve the bolt insertion sequence problem in Example 16.1.

EXAMPLE 16.1 Bolt Insertion Sequence Determination at 10 Locations

Solve the problem shown in Fig. 16-4 using a genetic algorithm.

Solution. The problem is solved by using a genetic algorithm (Huang and Arora, 1997). The population size N_p is set to 150 and I_g is set to 10. No seed designs are used for the problem. The optimum bolting sequence is not unique for the problem. With hole 1 as the starting point, the optimum sequence is determined as (1, 5, 4, 10, 7, 8, 9, 3, 6, 2) and the cost function value is 74.63 in. The number of function evaluations is 1445, which is much smaller than the total number of possibilities ($10! = 3,628,800$).

Two other problems are solved in Huang and Arora (1997). The first problem concerns determination of the bolting sequence for 16 locations. The optimum sequence is not unique for this example as well. The solution is obtained in 3358 function evaluations compared with the total number of possibilities, $16! \cong 2.092 \times 10^{13}$. The second example concerns the *A-pillar subassembly welding sequence* determination for a passenger vehicle. There are 14 welding locations. The objective is to determine the best welding sequence that minimizes the deformation at some critical points of the structure. Cases where one and two welding guns are available are considered. This is equivalent to having two salesmen traveling between N cities for the traveling salesman problem. The optimum sequences are obtained with 3341 and 3048 function evaluations for the two cases, which are much smaller than those for the full enumeration.

16.4 Applications

Numerous applications of genetic algorithms for different classes of problems have been presented in the literature. There are specialty conferences focusing on the development in the

genetic algorithms and their applications. The literature in this area is growing rapidly. Therefore a survey of all the applications is not attempted here. For the field of mechanical and structural design, some of the applications are covered in Arora (2002), Pezeshk and Camp (2002), Arora and Huang (1996), and Chen and Rajan (2000). Applications of the genetic algorithms for optimum design of electric transmission line structures are given in Kocer and Arora (1996, 1997, 1999, 2002).

Exercises for Chapter 16*

Solve the following problems using a genetic algorithm.

- 16.1 Example 15.1 with the available discrete values for the variables as $x_1 \in \{0, 1, 2, 3\}$, and $x_2 \in \{0, 1, 2, 3, 4, 5, 6\}$. Compare the solution with that obtained with the branch and bound method.
- 16.2 Exercise 3.34 using the outside diameter d_o and the inside diameter d_i as design variables. The outside diameter and thickness must be selected from the following available sets:

$$d_o \in \{0.020, 0.022, 0.024, \dots, 0.48, 0.50\} \text{ m}; \quad t \in \{5, 7, 9, \dots, 23, 25\} \text{ mm}$$

Check your solution using the graphical method of Chapter 3. Compare continuous and discrete solutions. Study the effect of reducing the number of elements in the available discrete sets.

- 16.3 Formulate the minimum mass tubular column problem described in Section 2.7 using the following data: $P = 100 \text{ kN}$, length, $l = 5 \text{ m}$, Young's modulus, $E = 210 \text{ GPa}$, allowable stress, $\sigma_a = 250 \text{ MPa}$, mass density, $\rho = 7850 \text{ kg/m}^3$, $R \leq 0.4 \text{ m}$, $t \leq 0.05 \text{ m}$, and $R, t \geq 0$. The design variables must be selected from the following sets:

$$R \in \{0.01, 0.012, 0.014, \dots, 0.38, 0.40\} \text{ m}; \quad t \in \{4, 6, 8, \dots, 48, 50\} \text{ mm}$$

Check your solution using the graphical method of Chapter 3. Compare continuous and discrete solutions. Study the effect of reducing the number of elements in the available discrete sets.

- 16.4 Consider the plate girder design problem described and formulated in Section 10.6. The design variables for the problem must be selected from the following sets

$$h, b \in \{0.30, 0.31, 0.32, \dots, 2.49, 2.50\} \text{ m}; \quad t_w, t_f \in \{10, 12, 14, \dots, 98, 100\} \text{ mm}$$

Compare the continuous and discrete solutions. Study the effect of reducing the number of elements in the available discrete sets.

- 16.5 Consider the plate girder design problem described and formulated in Section 10.6. The design variables for the problem must be selected from the following sets

$$h, b \in \{0.30, 0.32, 0.34, \dots, 2.48, 2.50\} \text{ m}; \quad t_w, t_f \in \{10, 14, 16, \dots, 96, 100\} \text{ mm}$$

Compare the continuous and discrete solutions. Study the effect of reducing the number of elements in the available discrete sets.

- 16.6 Solve problems of Exercises 16.4 and 16.5. Compare the two solutions, commenting on the effect of the size of the discreteness of variables on the optimum solution. Also, compare the continuous and discrete solutions.
- 16.7 Formulate the spring design problem described in Section 2.9 and solved in Section 13.5. Assume that the wire diameters are available in increments of 0.01 in, the coils can be fabricated in increments of $\frac{1}{16}$ th of an inch, and the number of coils must be an integer. Compare the continuous and discrete solutions. Study the effect of reducing the number of elements in the available discrete sets.
- 16.8 Formulate the spring design problem described in Section 2.9 and solved in Section 13.5. Assume that the wire diameters are available in increments of 0.015 in, the coils can be fabricated in increments of $\frac{1}{8}$ th of an inch, and the number of coils must be an integer. Compare the continuous and discrete solutions. Study the effect of reducing the number of elements in the available discrete sets.
- 16.9 Solve problems of Exercises 16.7 and 16.8. Compare the two solutions, commenting on the effect of the size of the discreteness of variables on the optimum solution. Also, compare the continuous and discrete solutions.
- 16.10 Formulate the problem of optimum design of prestressed concrete transmission poles described in Kocer and Arora (1996a). Compare your solution to that given in the reference.
- 16.11 Formulate the problem of optimum design of steel transmission poles described in Kocer and Arora (1996b). Solve the problem as a continuous variable optimization problem.
- 16.12 Formulate the problem of optimum design of steel transmission poles described in Kocer and Arora (1996b). Assume that the diameters can vary in increments of 0.5 in and the thicknesses can vary in increments of 0.05 in. Compare your solution to that given in the reference.
- 16.13 Formulate the problem of optimum design of steel transmission poles using standard sections described in Kocer and Arora (1997). Compare your solution to the solution given in the reference.
- 16.14 Formulate and solve three-bar truss of Exercise 3.50 as a discrete variable problem where the cross-sectional areas must be selected from the following discrete set:

$$A_i \in \{50, 100, 150, \dots, 4950, 5000\} \text{ mm}^2$$

Check your solution using the graphical method of Chapter 3. Compare continuous and discrete solutions. Study the effect of reducing the number of elements in the available discrete sets.

- 16.15 Solve Example 16.1 of bolt insertion sequence at 10 locations. Compare your solution to the one given in the example.
- 16.16 Solve the 16-bolt insertion sequence determination problem described in Huang and coworkers (1997). Compare your solution to the one given in the reference.

- 16.17 The material for the spring in Exercise 16.7 must be selected from one of three possible materials given in Table E16.17 (refer to Section 15.8 for more discussion of the problem) (Huang and Arora, 1997). Obtain a solution for the problem.

TABLE E16-17 Material Data for Spring Design Problem

	$G, \text{lb/in}^2$	$\rho, \text{lb-s}^2/\text{in}^4$	$\tau_a, \text{lb/in}^2$	U_p
Material type 1	11.5×10^6	7.38342×10^{-4}	80,000	1.0
Material type 2	12.6×10^6	8.51211×10^{-4}	86,000	1.1
Material type 3	13.7×10^6	9.71362×10^{-4}	87,000	1.5

G = shear modulus, ρ = mass density, τ_a = shear stress, U_p = relative unit price.

- 16.18 The material for the spring in Exercise 16.8 must be selected from one of three possible materials given in Table E16.17 (refer to Section 15.8 for more discussion of the problem) (Huang and Arora, 1997). Obtain a solution for the problem.

17 Multiobjective Optimum Design Concepts and Methods

Upon completion of this chapter, you will be able to:

- Explain basic terminology and concepts related to multiobjective optimization problems
- Explain the concepts of Pareto optimality and Pareto optimal set
- Solve your multiobjective optimization problem using a suitable formulation

Thus far, we have considered problems in which only one objective function needed to be optimized. However, there are many practical applications where the designer may want to optimize two or more objective functions simultaneously. These are called *multiobjective*, *multicriteria*, or *vector optimization* problems; we refer to them as *multiobjective optimization problems*. In this chapter, we describe basic terminology, concepts, and solution methods for such problems. The material is introductory in nature and is derived from Marler and Arora (2004) and many other references cited in there (e.g., Ehrgott and Granddibleaux, 2000).

17.1 Problem Definition

The general design optimization model defined in Chapter 2 is modified to treat multiobjective optimization problems as follows:

$$\text{minimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x}))$$

subject to

$$h_i(\mathbf{x}) = 0; i = 1 \text{ to } p$$

$$g_j(\mathbf{x}) \leq 0; j = 1 \text{ to } m \tag{17.1}$$

where k is the number of objective functions, p is the number of equality constraints, and m is the number of inequality constraints. $\mathbf{f}(\mathbf{x})$ is a k -dimensional vector of objective functions. Recall that the *feasible set* S (also called the *feasible design space*) is defined as a collection of all the feasible design points, as

$$S = \{\mathbf{x} | h_i(\mathbf{x}) \leq 0; i = 1 \text{ to } p; \text{ and } g_j(\mathbf{x}) \leq 0; j = 1 \text{ to } m\}$$

The problem shown in Eq. (17.1) usually does not have a unique solution, and this idea is illustrated by contrasting single-objective and multiobjective problems. *Note that we shall use the terms “cost function” and “objective function” interchangeably in this chapter.* Examples 17.1 and 17.2 illustrate the basic difference between single-objective and multi-objective optimization problems.

EXAMPLE 17.1 Single-Objective Optimization Problem

$$\text{minimize } f_1(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 5)^2 \quad (\text{a})$$

$$\text{subject to } g_1 = -x_1 - x_2 + 10 \leq 0 \quad (\text{b})$$

$$g_2 = -2x_1 + 3x_2 - 10 \leq 0 \quad (\text{c})$$

Solution. Figure 17-1 shows a graphical representation for the problem. The feasible set S is convex as shown in the figure. A few objective function contours are also shown. It is seen that the problem has a distinct minimum at the point A (4, 6) with the objective function value of $f_1(4, 6) = 5$. At the minimum point, both constraints are active. Note that since the objective function is also strictly convex, point A represents the unique global minimum for the problem.

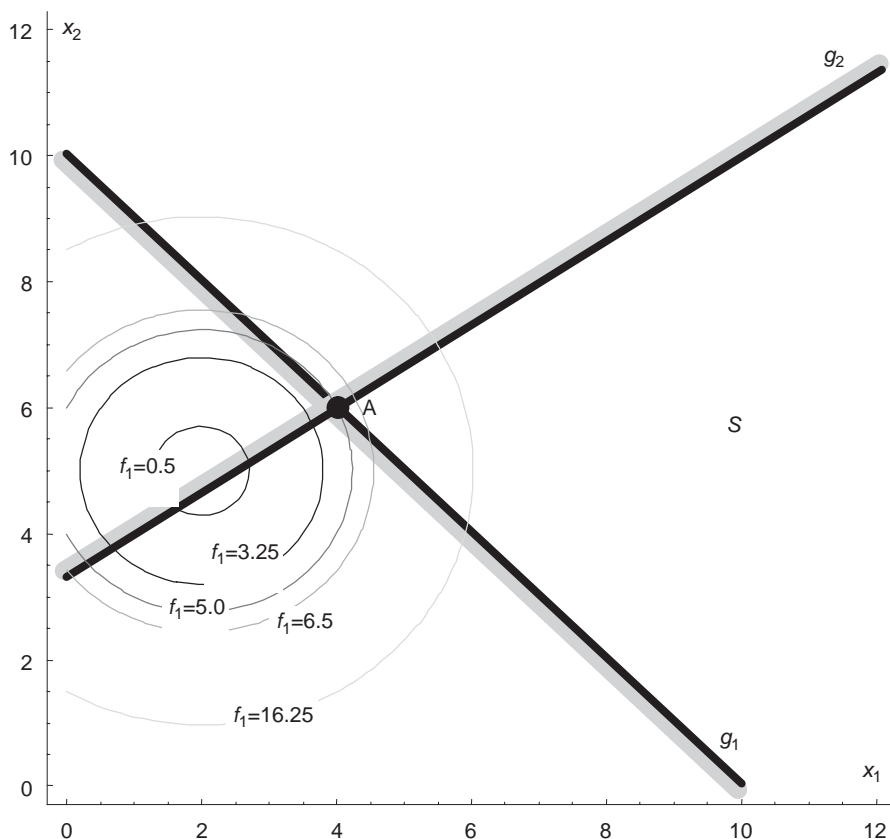


FIGURE 17-1 Graphical representation of a single-objective optimization problem.

EXAMPLE 17.2 Two-Objective Optimization Problem

A second objective function is added to the Example 17.1 to obtain the following two-objective problem:

$$\text{minimize } f_1(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 5)^2 \quad (\text{a})$$

$$f_2(\mathbf{x}) = (x_1 - 4.5)^2 + (x_2 - 8.5)^2 \quad (\text{b})$$

subject to same constraints as for Example 17.1.

Solution. Figure 17-2 is a modification of Fig. 17-1, where contours of the second objective function are also shown. The minimum value of f_2 is 3.25 at point B (5.5, 7.0). Note that f_2 is also a strictly convex function, and so point B is a unique global minimum point for f_2 . The minimum points for the two objective functions are different. Therefore, if one wishes to minimize f_1 and f_2 simultaneously, then pinpointing a single optimum point is not possible. In fact, there are *infinitely many possible solution points called the Pareto optimal set*, which is explained later. The challenge is to find a solution that suits one's requirements. This dilemma requires the description of additional terminology and solution concepts.

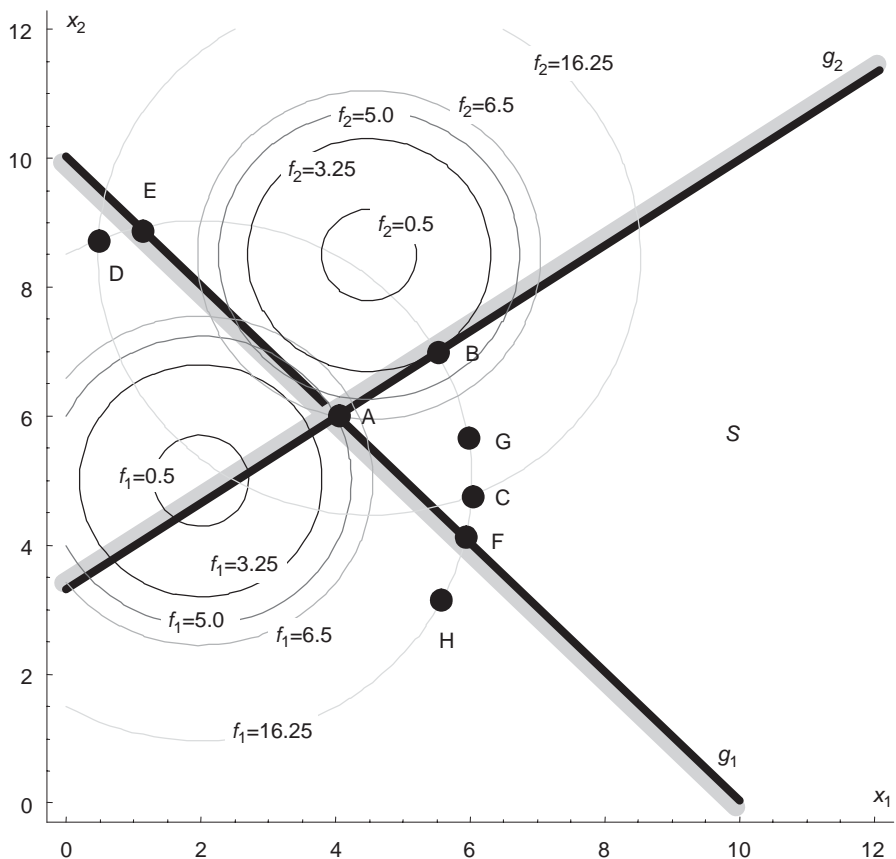


FIGURE 17-2 Graphical representation of a two-objective optimization problem.

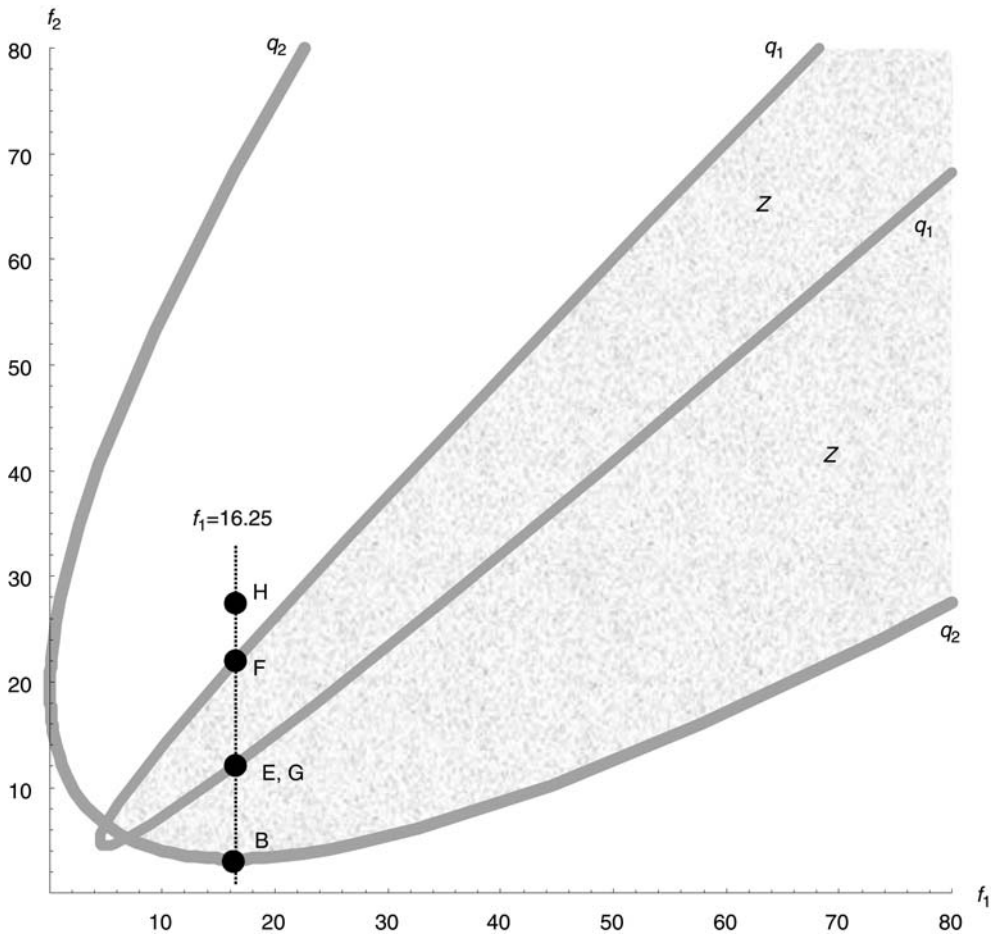


FIGURE 17-3 Graphical representation of a two-objective optimization problem in the criterion space.

17.2 Terminology and Basic Concepts

17.2.1 Criterion Space and Design Space

Example 17.2 is depicted in the *design space* in Fig. 17-2. That is, the constraints g_1 and g_2 , and the objective function contours are plotted as functions of the design variables x_1 and x_2 . Alternatively, a multiobjective optimization problem may also be depicted in the *criterion space* (also called the *cost space*), where the axes represent different objective functions. For the present problem, f_1 and f_2 are the axes in the criterion space, as shown in Figs. 17-3 and 17-4. q_1 represents the g_1 boundary, and q_2 represents the g_2 boundary. In general, a curve in the design space in the form $g_j(\mathbf{x}) = 0$ is translated into a curve q_j in the criterion space simply by evaluating the values of the objective functions at different points on the constraint curve in the design space. The feasible criterion space Z is defined simply as the set of objective function values corresponding to the feasible points in the design space; i.e.,

$$Z = \{\mathbf{f}(\mathbf{x}) | \mathbf{x} \text{ in the feasible set } S\}.$$

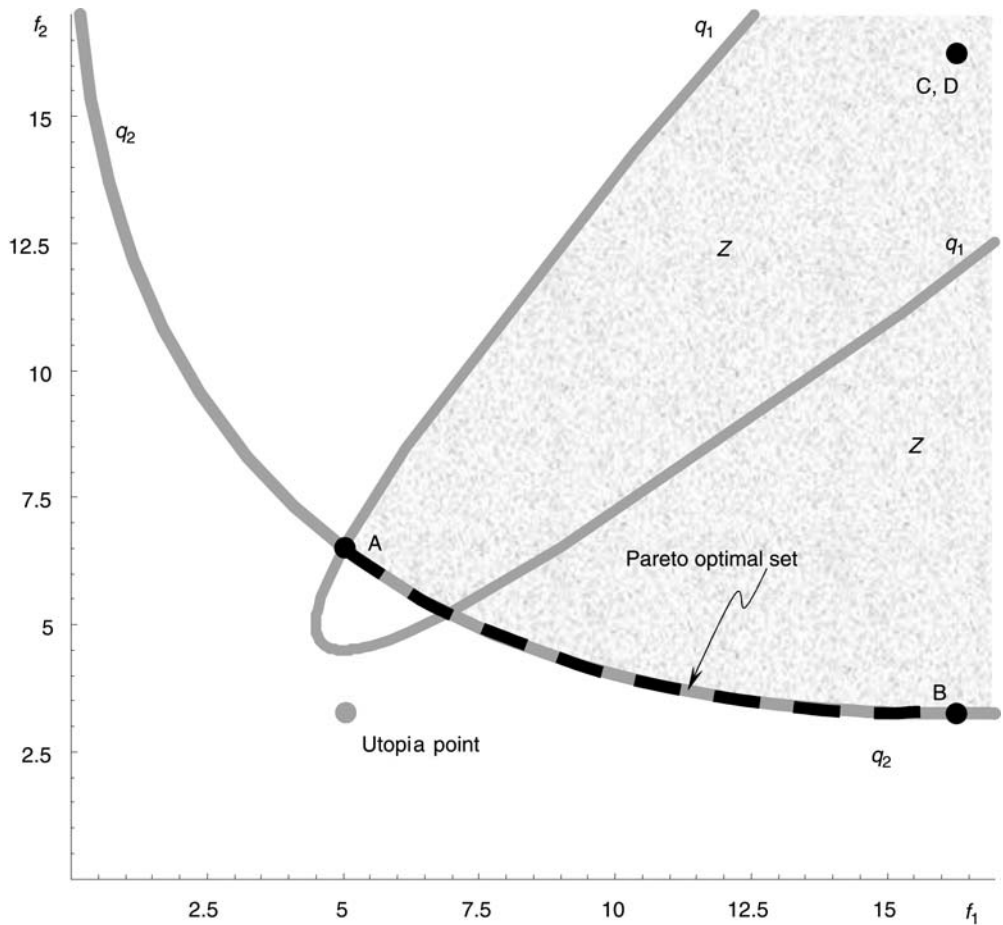


FIGURE 17-4 Illustration of Pareto optimal set and utopia point in the criterion space.

The feasible points in the design space map onto a set of points in the criterion space. Note that although q_j represents g_j in the criterion space, it may not necessarily represent the boundaries of the *feasible criterion space*. This is seen in Fig. 17-3 where the feasible criterion space for the problem of Example 17.2 is displayed. All portions of the curves q_1 and q_2 do not form boundaries of the feasible criterion space. This concept of feasible criterion space is important and used frequently, so we will discuss it further.

Let us first consider the single-objective function problem depicted in Fig. 17-1. The feasible criterion space for the problem is the line f_1 that starts at 5, the minimum value for the function, and goes to infinity. Note that each feasible design point corresponds only to one objective function value; it maps onto only one point on the feasible criterion line. However, for one objective function value, there may be many different feasible design points in the feasible design space S . For instance, in Fig. 17-1, there are infinitely many design points that result in $f_1 = 16.25$ as seen for the contour $f_1 = 16.25$. Note also that the contour $f_1 = 16.25$ passes through the infeasible region as well. Thus, for a given objective function value (a given point in the feasible criterion space), there can be feasible or infeasible points in the design space. Note also that the objective function values for design points on the constraint boundaries for g_1 and g_2 fall on the line f_1 in the criterion space.

Now let us consider the problem with two objective functions and study the relationship between constraint boundaries in the design space and the corresponding curves in the criterion space. Let us consider two feasible points E and F in the criterion space as shown in Fig. 17-3. Both points are on the curve q_1 and have a value of 16.25 for f_1 . Since both points are on the curve q_1 , they must also lie on the constraint boundary g_1 in the design space. They indeed are on the g_1 line in the design space at E(1.076, 8.924) and F(5.924, 4.076), as shown in Fig. 17-2. Whereas both the points satisfy the constraint g_1 , the point E violates the constraint g_2 , and thus, is not in the feasible set S . Then the question is how can point E be in the feasible criterion space? It turns out that there is another point G on the $f_1 = 16.25$ contour in Fig. 17-2 that is feasible and has the same value for f_2 as for the point E. Therefore point G also maps onto point E in the criterion space as shown in Fig. 17-3. Thus, a feasible point in the criterion space can map onto multiple points in the design space, some of which may violate constraints. Note that the feasible point C and infeasible point D in Fig. 17-2 both map onto a feasible point C in the criterion space, as seen in Fig. 17-4. An infeasible point H in the criterion space in Fig. 17-3 having $f_1 = 16.25$ maps onto an infeasible point H in the design space, as seen in Fig. 17-2. Thus, the feasible criterion space indicates all the points in the criterion space that are obtained by using feasible points in the design space.

A concept that is related to the feasibility of design points is that of *attainability*. 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 feasible design space. Whereas each point in the feasible design space translates to a point in the criterion space, the reverse may not be true; i.e., every point in the criterion space does not necessarily correspond to a single point \mathbf{x} in the feasible design space S , as we saw in the foregoing example. Thus, even with an unconstrained problem, only certain points in the criterion space are *attainable*. We shall use the symbol Z to indicate points in the criterion space that are attainable and correspond to a feasible point in the set S . The set Z is also referred to as the *attainable set*. For the relatively simple problem in Example 17.2, it is possible to depict Z , as illustrated in Figs. 17-3 and 17-4, but generally, it is not possible to depict the feasible criterion space directly.

As noted earlier for Example 17.1, the feasible criterion space is the real line starting at 5 and going up to infinity. Therefore, only the objective function values of 5 and higher are *attainable* and constitute the feasible criterion space Z for the problem.

17.2.2 Solution Concepts

From a classical standpoint, optimizing a single function simply entails determining a set of stationary points, identifying a local maximum or minimum, and possibly finding the global optimum, such as point A in Fig. 17-1. In contrast, the process of determining a solution for a multiobjective optimization problem is slightly more complex and less definite than that for a single-objective problem. As seen for the Example 17.2 and depicted in Fig. 17-2, point A is the minimum for f_1 and point B is the minimum for f_2 . But, which design point minimizes both f_1 and f_2 simultaneously? This is not clear even for this simple problem. Therefore, it is not clear what is meant by the minimum of multiple functions that may have opposing characteristics since what decreases the value of one function may increase the value of another. Therefore, in this section we describe some solution concepts related to multiobjective optimization problems.

Pareto Optimality The predominant solution concept in defining solutions for multiobjective optimization problems is that of *Pareto optimality* (Pareto, 1906). A point \mathbf{x}^* in the feasible design space S is called Pareto optimal if there is no other point \mathbf{x} in the set S that reduces at least one objective function without increasing another one. This is defined more precisely as follows:

A point \mathbf{x}^* in the feasible design space S is *Pareto optimal* if and only if there does not exist another point \mathbf{x} in the set S such that $\mathbf{f}(\mathbf{x}) \leq \mathbf{f}(\mathbf{x}^*)$ with at least one $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$.

Note that inequalities between vectors apply to every component of each vector; e.g., $\mathbf{f}(\mathbf{x}) \leq \mathbf{f}(\mathbf{x}^*)$ implies $f_1 \leq f_1^*, f_2 \leq f_2^*$, and so on. The set of all Pareto optimal points is called the *Pareto optimal set*, and this term can refer to points in the design space or to points in the criterion space. The above definition means that for \mathbf{x}^* to be called the Pareto optimal point, no other point exists in the feasible design space S that improves at least one objective function while keeping others unchanged.

As an example of a Pareto optimal point, consider point A in Figs. 17-2 and 17-4. It is not possible to move from this point and simultaneously decrease the value of f_1 and f_2 without violating a constraint, i.e., without moving into the infeasible region. Therefore, point A is a Pareto optimal point. However, it is possible to move from point C and simultaneously reduce the values of both f_1 and f_2 . This can be seen most clearly in Fig. 17-4. Thus, point C is not Pareto optimal. In Fig. 17-2, the points on the line between A and B along g_2 represent the Pareto optimal set. In Fig. 17-4, the Pareto optimal set is shown as the curve between points A and B along q_2 . In fact, *the Pareto optimal set is always on the boundary of the feasible criterion space Z*. When there are just two objective functions, as with this example, then the minimum points of each objective function define the endpoints of the Pareto optimal curve, assuming the minima to be unique.

Note that although the Pareto optimal set is always on the boundary of Z , it is not necessarily defined by the constraints. As noted earlier, Z exists even for unconstrained problems. In such a case, the Pareto optimal set is defined by the relationship between the gradients of the objective functions. In the simple case *when there are just two objective functions, the gradients of the functions point in opposite directions at all Pareto optimal points*. An exception to this rule is the individual minimum points for the functions at which the gradient is zero. For Example 17.2 without the constraints g_1 and g_2 , the Pareto optimal set is along the line connecting the centers of the circles for the two objective functions, i.e., points (2, 5) and (4.5, 8.5). This line would map onto a curve in the criterion space in Fig. 17-3.

Weak Pareto Optimality A concept closely related to Pareto optimality is that of weak Pareto optimality. At the weakly Pareto optimal points, it is possible to improve some objective functions without penalizing others. A *weakly Pareto optimal* point is defined as follows:

A point \mathbf{x}^* in the feasible design space S is weakly Pareto optimal if and only if there does not exist another point \mathbf{x} in the set S such that $\mathbf{f}(\mathbf{x}) < \mathbf{f}(\mathbf{x}^*)$. That is, there is no point that improves all the objective functions simultaneously; however, there may be points that improve some of the objectives while keeping others unchanged.

In contrast to weakly Pareto optimal points, no objective function can be improved from a Pareto optimal point without detriment to another objective function. It will be seen later that there are numerical algorithms for multiobjective optimization that may converge to weakly Pareto optimal solutions as opposed to always giving Pareto optimal solutions.

Efficiency and Dominance Efficiency is another primary concept in multiobjective optimization and is defined as follows:

A point \mathbf{x}^* in the feasible design space S is efficient if and only if there does not exist another point \mathbf{x} in the set S such that $\mathbf{f}(\mathbf{x}) \leq \mathbf{f}(\mathbf{x}^*)$ with at least one $f_i(\mathbf{x}) < f_i(\mathbf{x}^*)$. Otherwise, \mathbf{x}^* is inefficient. The set of all efficient points is called the *efficient frontier*.

Another common concept is that of *nondominated* and *dominated* points defined as follows:

A vector of objective functions $\mathbf{f}^* = \mathbf{f}(\mathbf{x}^*)$ in the feasible criterion space Z is nondominated if and only if there does not exist another vector \mathbf{f} in the set Z such that $\mathbf{f} \leq \mathbf{f}^*$, with at least one $f_i < f_i^*$. Otherwise, \mathbf{f}^* is dominated.

Note that the definitions of Pareto optimal and efficient points are the same. Also, the definitions of efficient and nondominated points are similar. The only distinction is that efficiency refers to points in the design space and nondominance refers to the points in the criterion space. Pareto optimality, however, 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. Pareto optimality, on the other hand, implies a condition in terms of the complete feasible design or criterion space. Genetic algorithms and some random search methods for multiobjective optimization update and store a discrete set of potential solution points in each iteration. Each new addition to this set is compared with all the objective function values of potential solution points to determine if the new point is dominated. If it is nondominated, then it is kept in the set of potential solution points; note, however, that this point may not be Pareto optimal.

Utopia Point This is a unique point in the criterion space that is defined as follows:

A point \mathbf{f}° in the criterion space is called the utopia point if $f_i^\circ = \min\{f_i(\mathbf{x}) \mid \text{for all } \mathbf{x} \text{ in the set } S\}$, $i = 1$ to k . It is also called the *ideal point*.

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. 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 the objective functions. Thus, *the utopia point exists only in the criterion space and, in general, it is not attainable*.

Figure 17-4 shows the Pareto optimal set and the utopia point for the problem of Example 17.2. The Pareto optimal set is on the boundary of Z and coincides with the curve q_2 . The utopia point is located at the point (5, 3.25), as calculated before. Note that the utopia point is not in Z , and is therefore unattainable.

Compromise Solution The next best thing to a utopia point is a solution that is as close as possible to the utopia point. Such a solution is called a *compromise solution*. 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:

$$D(\mathbf{x}) = \|\mathbf{f}(\mathbf{x}) - \mathbf{f}^\circ\| = \left\{ \sum_{i=1}^k [f_i(\mathbf{x}) - f_i^\circ]^2 \right\}^{1/2} \quad (17.2)$$

where f_i° represents a component of the utopia point in the criterion space. *Compromise solutions are Pareto optimal*.

17.2.3 Preferences and Utility Functions

Because mathematically, there are infinitely many Pareto optimal solutions, one often has to make decisions concerning which solution is preferred. Fundamentally, this specification of preferences is based on opinions concerning points in the criterion space. Ideally, a multi-objective optimization method should reflect the user's preferences, if known; i.e., it should incorporate how the user feels about different solution points. However, having a mathematical model or algorithm to represent one's preferences perfectly is usually impossible. Nonetheless, different methods for multiobjective optimization try to incorporate preferences in different ways. This idea of accurately incorporating and reflecting preferences is a common and significant issue for multiobjective optimization methods. Consequently, some work has been done to develop methods that effectively incorporate preferences. These methods typically try to capture knowledge about the problem functions and incorporate it into mathematical expressions that are then used in multiobjective optimization methods. One such recent method that captures this knowledge quite accurately is called *physical programming*. The method has been used successfully in several applications (Messac, 1996; Chen *et al.*, 2000, Messac *et al.*, 2001; Messac and Mattson, 2002).

Essentially, there are three approaches for expressing preferences about different objective functions. Preferences can be declared before solving the multiobjective optimization problem. For instance, one may specify weights associated with each objective, indicating the relative importance of each objective. Alternatively, preferences can be indicated by interacting with the optimization routine and making choices based on intermediate optimization results. For engineering applications, such approaches can be awkward, especially with problems that require a significant amount of time to evaluate the problem functions. Finally, it is possible to calculate the complete Pareto optimal set (or its approximation) and then select a single solution point after the problem has been solved. This, however, is not practical for more than three objective functions (although selected subsets of two or three objectives functions may be displayed). In some instances, the decision maker may not be able to concretely define preferences. Thus, as a special case, one may choose not to declare preferences at all.

A *utility function* is a mathematical expression that attempts to model the decision maker's preferences. A utility function is most relevant to methods that indicate preferences before the problem is solved. In this context, utility, which is modeled with a utility function, represents an individual's degree of contentment. Utility emphasizes a decision maker's satisfaction, which is slightly different from the usual meaning of usefulness or worth. The utility function is a scalar function incorporating various objective functions.

17.2.4 Vector Methods and Scalarization Methods

One of the predominant classifications for multiobjective approaches is known as *scalarization* and *vector optimization methods*. With scalarization methods, the components of objective function vector are combined to form a scalar objective function. Then, one can use standard single-objective methods to optimize the resulting scalar function. Alternatively, the term "vector optimization" implies that each objective function is treated independently. We shall describe examples of both approaches.

17.2.5 Generation of Pareto Optimal Set

A key characteristic of multiobjective optimization methods 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 the Pareto optimal points. Alternatively, other methods are able to capture all the points in the Pareto optimal set (when the problem is solved by changing the method parameters) but may also provide non-Pareto optimal points. The former quality is beneficial when one is interested in using a method to obtain just one solution point. The latter quality is useful when the complete Pareto optimal

set needs to be generated. We shall note this feature of each method when it is described in later sections. The tendency of a particular method to result in non-Pareto optimal points and the ability of a method to capture all the Pareto optimal points depend not only on the method itself but also on the nature of the problem being solved.

17.2.6 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 many approaches, the most robust approach is to normalize the objective functions as follows:

$$f_i^{\text{norm}} = \frac{f_i(x) - f_i^\circ}{f_i^{\text{max}} - f_i^\circ} \quad (17.3)$$

$f_i^{\text{norm}}(\mathbf{x})$ generally has values between 0 and 1, depending on the accuracy and method with which $f_i^{\text{max}}(\mathbf{x})$ and $f_i^\circ(\mathbf{x})$ are determined. There are two approaches for determining $f_i^{\text{max}}(\mathbf{x})$. One can define $f_i^{\text{max}}(\mathbf{x})$ such that $f_i^{\text{max}}(\mathbf{x}) = \max_{1 \leq j \leq k} f_j(\mathbf{x}_j^*)$, where \mathbf{x}_j^* is the point that minimizes the j th objective function. This implies that each objective $f_j(\mathbf{x})$ needs to be minimized to determine \mathbf{x}_j^* . Then all objective functions need to be evaluated at \mathbf{x}_j^* . The maximum of all the f_j values is $f_i^{\text{max}}(\mathbf{x})$. This process also determines the utopia point $f_i^\circ(\mathbf{x})$. It is noted that this normalization process may not be practical in some cases. Therefore, instead of $f_i^{\text{max}}(\mathbf{x})$, one may use the absolute maximum value of $f_i(\mathbf{x})$, or its approximation based on engineering intuition. Similarly, the utopia point may be replaced with a reasonable estimate (called the *aspiration point*, *target value*, or *goal*). We shall assume that the objective functions have been normalized. Note, however, that if all the objective functions have similar values, normalization may not be needed.

17.2.7 Optimization Engine

Most approaches for solving multiobjective optimization problems actually entail formulating the multiple objective functions into a single-objective problem or a series of problems. Then, a standard single-objective optimization routine is used to solve the consequent formulation. We call this routine the *optimization engine*. The performance of most multi-objective methods depends on which optimization engine is used.

17.3 Multiobjective Genetic Algorithms

Genetic algorithms (GAs) for single-objective optimization can be extended to provide an effective approach for solving multiobjective optimization problems as well. Since GAs for multiobjective optimization build upon the GAs for single-objective optimization, the concepts and procedures described previously in Chapter 16 should be reviewed.

Because genetic algorithms do not require gradient information, they can be effective regardless of the nature of the problem functions. They combine the use of random numbers and information from previous iterations to evaluate and improve a population of points (a group of potential solutions) rather than a single point at a time. Another appeal of genetic algorithms is their *ability to converge to the Pareto optimal set* rather than a single Pareto optimal point (Osyczka, 2002).

Although the algorithms in this section are intended for application to engineering problems, much of the literature uses terminology from the fields of biology and genetics. Thus,

for the sake of clarity, basic definitions from Chapter 16 are reviewed here and some new terms are introduced.

A *population* represents a set of design points in the design space. A *subpopulation* is a subset of points in a generation. The *generation* refers to a computational iteration. To say that a *point survives* into the next generation means that the point is *selected* for use in the next iteration. A *niche* is a group of points that are close together (typically in terms of distance in the criterion space).

Multiobjective Genetic Algorithms The primary questions when developing genetic algorithms for multiobjective problems are: how to evaluate fitness, how to incorporate the idea of Pareto optimality, and how to determine which potential solution points should be selected (survive) for the next iteration (generation). Note that the fitness of a design point (determined usually by a fitness function) is used in the selection process; i.e., to decide whether to include the design in the next generation. However, in some multiobjective genetic algorithms, fitness of a design is neither defined nor used; instead some selection strategy is used directly to select the designs for the next iteration. The approaches that are described in this section, collectively address these two issues. Different selection techniques are discussed that serve as potential ingredients in a genetic multiobjective optimization algorithm. Once a set of designs is selected for the next generation, the cross-over and mutation operators, described in Chapter 16, are used to create a new set of designs that are subjected to the selection process again; thus, the iterations continue this way.

In general, genetic algorithms for multiobjective optimization are still evolving. We shall describe some basic ideas and techniques that can be combined, modified, and used in different ways in a specific genetic algorithm for selection of designs for the next generation.

Vector Evaluated Genetic Algorithm One of the first treatments of multiobjective genetic algorithms was presented by Schaffer (1985), which has provided a foundation for later developments. The general idea behind the approach, called the *vector evaluated genetic algorithm* (VEGA), involves producing smaller subsets (subpopulations) of the current designs (population) in a given iteration (generation). One subset is created by evaluating one objective function at a time rather than aggregating all the functions. The *selection process* is composed of a series of computational loops, and during each loop, the fitness of each member of the current set of designs is evaluated using a single-objective function. Then, certain members of the population are selected and passed on to the next generation using the stochastic processes discussed earlier in Chapter 16. This selection process is repeated for each objective function. Consequently, for a problem with k objectives, k subsets are created, each with N_p/k members, where N_p is the size of the entire set (population size). The resulting subsets are then combined to yield a new population.

This process is based on the idea that the minimum of a single-objective function is a Pareto optimal point (assuming the minimum is unique). Such minima generally define vertices of the Pareto optimal set. Consequently, Schaffer's method does not yield an even distribution of Pareto optimal points. Solutions in a given generation tend to cluster around individual function minima. This is analogous to the evolution of species, where a *species* is a class of organisms with common attributes.

Ranking A class of alternatives to VEGA, when it comes to evaluating fitness and selecting designs for the next generation, involves giving each design a rank based on whether it is dominated in the criterion space (Goldberg, 1989; Srinivas and Deb, 1995; Cheng and Li, 1998). Fitness then is based on a design's rank within a population. The means of determining rank and assigning fitness values associated with rank may vary from method to method, but the general approach is common as described in the following discussion.

For a given set of designs, the objective functions are evaluated at each point. All non-dominated points receive a rank of 1. Determining whether a point is dominated (performing a *nondominated check*) entails comparing the vector of objective function values at the point with the vector at all other points. Then, the points with a rank of 1 are temporarily removed from consideration, and the points that are nondominated relative to the remaining group are given a rank of 2. This process is repeated until all points are ranked. Those points with the lowest rank have the highest fitness value. That is, fitness is determined such that it is inversely proportional to the rank.

Pareto Fitness Function The fitness function for a problem with multiple objectives can be defined in many different ways (Balling *et al.*, 1999, 2000; Balling, 2000). The following function, called the *maximin fitness function*, has been used successfully in some applications (Balling, 2003):

$$F(\mathbf{x}_i) = \max_{j \neq i; j \in P} \left[\min_{1 \leq s \leq k} \{f_s(\mathbf{x}_i) - f_s(\mathbf{x}_j)\} \right] \quad (17.4)$$

Here, $F(\mathbf{x}_i)$ is the fitness of the i th design, P is the set of nondominated points in the current population, and it is assumed that each objective function has been scaled by dividing it by an appropriate positive constant. Thus, with each iteration, one must first determine all the nondominated points before evaluating the fitness of the designs. Note that the nondominated points have negative fitness values. This fitness function automatically penalizes clustering of nondominated points. Thus, compared with other selection approaches, this approach is relatively simple and effective.

Pareto-Set Filter It is possible to have a Pareto optimal point in a particular iteration that does not appear in subsequent iterations; that is, it may get dropped from further consideration during the selection process. To guard against this situation, a *Pareto-set filter* can be used. Regardless of how fitness is determined, most genetic multiobjective optimization methods incorporate some type of Pareto-set filter to avoid losing potential Pareto optimal solutions, which is described as follows (Cheng and Li, 1997). Basically, the algorithm stores two sets of solutions: the current population and the *filter* (another set of potential solutions). The filter is called an *approximate Pareto set*, and it provides an approximation of the Pareto optimal set. With each iteration, points with a rank of 1 are saved in the filter. When new points from subsequent iterations are added to the filter, they are subjected to a nondominated check within the filter, and the dominated points are discarded. The capacity of the filter typically is set to the size of the population. When the filter is full, points at a minimum distance from other points are discarded in order to maintain an even distribution of Pareto optimal points. The filter eventually converges on the true Pareto optimal set.

Elitist Strategy Although this procedure is similar to the Pareto-set filter approach, it provides an alternative means for ensuring that Pareto optimal solutions are not lost (Ishibuchi and Murata, 1996; Murata *et al.*, 1996). It functions independently of the ranking scheme. As with the Pareto-set filter, two sets of solutions are stored: a current population and a *tentative set of nondominated solutions*, which is an approximate Pareto optimal set. With 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, the k solutions with the best values for each objective function can be regarded as elite points and preserved for the next generation.

Tournament Selection This is another technique for selecting designs that are used in subsequent iterations. Although this technique concerns the selection process, it circumvents the idea of fitness. It is an alternative to the ranking approach described above and proceeds as follows (Horn *et al.*, 1994; Srinivas and Deb, 1995). Two points, called *candidate points*, are randomly selected from the current population. These candidate points essentially compete for survival in the next generation. A separate set of points called a *tournament set* or *comparison set* also is randomly compiled. The candidate points are then compared with each member of the tournament set. If both points are dominated by the points in the tournament set, then another pair is selected. If there is only one candidate that is nondominated relative to the tournament set, that candidate is selected for use in the next iteration. However, if there is no preference between candidates, or if there is a tie, *fitness sharing* (explained later) is used to select a candidate. The size of the tournament set is prespecified as a percentage of the total population. The size of the tournament set imposes the degree of difficulty in surviving, which is called the *domination pressure*. An insufficient number of Pareto optimal points will be found if the tournament size is too small, and premature convergence may result if the tournament size is too large.

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* or *niche-formation methods*) are methods for ensuring that a set of designs does not converge to a niche, i.e., a limited number of Pareto optimal points. 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 drift* (or *population drift*), and 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 (Deb, 1989; Fonseca and Fleming, 1993; Horn *et al.*, 1994; Srinivas and Deb, 1995; Narayana and Azarm, 1999). 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. This way the fitness of all the points in a niche is shared in some sense, and thus the term “fitness sharing.”

In reference to tournament selection, when two candidates are both either nondominated or dominated, the most fit candidate is the one with the least number of individuals surrounding it (within a specified distance in the criterion space). This is called *equivalence class sharing*.

17.4 Weighted Sum Method

The most common approach to multiobjective optimization is the *weighted sum* method:

$$U = \sum_{i=1}^k w_i f_i(\mathbf{x}) \quad (17.5)$$

Here, \mathbf{w} is a vector of weights typically set by the decision maker such that $\sum_{i=1}^k w_i = 1$ and $\mathbf{w} > \mathbf{0}$. If objectives are not normalized, w_i 's need not add to 1. As with most methods that involve objective function weights, setting one or more of the weights to zero can result in weakly Pareto optimal points. The relative value of the weights generally reflects the relative importance of the objectives. This is another common characteristic of weighted methods. If all the weights are omitted or are set to 1, then all objectives are treated equally.

The weights can be used in two ways. The user may either set \mathbf{w} to reflect preferences before the problem is solved or systematically alter \mathbf{w} to yield different Pareto optimal points (generate the Pareto optimal set). In fact, *most methods that involve weights can be used in both of these capacities*; i.e., to generate a single solution or multiple solutions.

This method is easy to use, and if all the weights are positive, the minimum of Eq. (17.5) is always Pareto optimal. However, there are a few recognized difficulties with the weighted sum method (Koski, 1985; Das and Dennis, 1997). First, even with the use of some of the methods discussed in the literature for determining weights, a satisfactory a priori selection of weights does not necessarily guarantee that the final solution will be acceptable; one may have to re-solve the problem with new 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. Although nonconvex Pareto optimal sets are relatively uncommon, some examples are noted in the literature (Koski, 1985; Stadler and Dauer, 1992; Stadler, 1995). The final 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.

17.5 Weighted Min-Max Method

The *weighted min-max* method (also called the *weighted Tchebycheff* method) is formulated as follows:

$$U = \max_i \{w_i [f_i(\mathbf{x}) - f_i^o]\} \quad (17.6)$$

A common approach for treatment of Eq. (17.6) is to introduce an additional unknown parameter λ as follows: minimize λ subject to additional constraints

$$w_i [f_i(\mathbf{x}) - f_i^o] - \lambda \leq 0; i = 1 \text{ to } k \quad (17.7)$$

Whereas the weighted sum method of Section 17.4 always yields Pareto optimal points but may miss certain points when the weights are varied, this method can provide all the Pareto optimal points (the complete Pareto optimal set). However, it may provide non-Pareto optimal points as well. Nonetheless, the solution to the min-max approach is always weakly Pareto optimal, and if the solution is unique, then it is Pareto optimal.

Advantages of the method are: (1) it provides a clear interpretation of minimizing the largest difference between $f_i(\mathbf{x})$ and f_i^o , (2) it can provide all the Pareto optimal points, (3) it always provides a weakly Pareto optimal solution, and (4) it is relatively well suited for generating the complete Pareto optimal set (with variation in the weights). Disadvantages are: (1) it requires the minimization of each objective when using the utopia point, which can be computationally expensive, (2) it requires that additional constraints be included, and (3) it is not clear exactly how to set the weights when only one solution point is desired.

17.6 Weighted Global Criterion Method

This is a scalarization method that combines all objective functions to form a single function. Although the term “global criterion” can refer to any scalarized function, it has been used in the literature primarily for formulations similar to the ones presented in this section. Although a global criterion may be a mathematical function with no correlation to prefer-

ences, a *weighted* global criterion is a type of utility function in which parameters are used to model preferences.

The most common weighted global criterion is defined as follows:

$$U = \left\{ \sum_{i=1}^k [w_i (f_i(\mathbf{x}) - f_i^\circ)]^p \right\}^{1/p} \quad (17.8)$$

Solutions using the global criterion formulation depend on the values of both \mathbf{w} and p . Generally, p is proportional to the amount of emphasis placed on minimizing the function with the largest difference between $f_i(\mathbf{x})$ and f_i° . The root $1/p$ may be omitted because the formulations with and without the root theoretically provide the same solution. p and \mathbf{w} typically are not varied or determined in unison. Rather, a fixed value for p is selected, and then either \mathbf{w} is selected to reflect preferences before the problem is solved or it is systematically altered to yield different Pareto optimal points.

Depending on how p is set, the global criteria can be reduced to other common methods. For instance, when $p = 1$, Eq. (17.8) is similar to a weighted sum with the objective functions adjusted with the utopia point. When $p = 2$ and weights equal to 1, Eq. (17.8) represents the distance from the utopia point, and the solution usually is called a compromise solution, as discussed earlier. When $p = \infty$, Eq. (17.8) reduces to Eq. (17.6). With the weighted global criterion method, increasing the value of p can increase its effectiveness in providing the complete Pareto optimal set (Athans and Papalambros, 1996; Messac *et al.*, 2000a,b). This explains why the weighted min-max approach can provide the complete Pareto optimal set with variation in the weights; the weighted min-max method shown in Eq. (17.6) is the limit of Eq. (17.8) as $p \rightarrow \infty$.

For computational efficiency or in cases where a function's independent minimum may be difficult to determine, *one may approximate the utopia point* with \mathbf{z} , which is called an *aspiration point*, *reference point*, *goal*, or *target point*. When this is done, U is called an *achievement function*. Then, the user has three different parameters that can be used to specify different types of preferences: \mathbf{w} , p , and \mathbf{z} . Assuming \mathbf{w} is fixed, then every Pareto optimal point may be captured by using a different aspiration point \mathbf{z} , as long as the aspiration point is not in the feasible criterion space Z . However, this is not a practical approach for generating the complete Pareto optimal set. Often, it is not possible to determine whether \mathbf{z} is in the feasible criterion space Z before solving the problem. In addition, if the aspiration point is in the feasible criterion space, the method may provide non-Pareto optimal solutions. Thus, it is recommended that the utopia point be used whenever possible. In addition, the aspiration point should not be varied as a parameter of the method but only as an approximation of the utopia point.

Equation (17.8) always yields a Pareto optimal solution as long as $\mathbf{w} > \mathbf{0}$ and as long as the utopia point is used. However, it may skip certain Pareto optimal points, depending on the nature of the objective functions and the value of p that is used. Generally, using a higher value for p enables one to better capture all Pareto optimal points (with variation in \mathbf{w}).

One can view the arguments of the summation in Eq. (17.8) in two ways: as transformations of the original objective functions or as components of a distance function that minimizes the distance between the solution point and the utopia point in the criterion space. Consequently, global criterion methods often are called *utopia point methods* or *compromise programming* methods, as the decision maker usually has to compromise between the final solution and the utopia point.

Advantages of the global criterion method are: (1) it gives a clear interpretation of minimizing the distance from the utopia point (or, aspiration point), (2) it gives a general

formulation that reduces to many other approaches, (3) it allows multiple parameters to be set to reflect preferences, and (4) it always provides a Pareto optimal solution when the utopia point is used. *Disadvantages* of the method are: (1) the use of the utopia point requires minimization of each objective function, which can be computationally expensive, (2) the use of an aspiration point requires that it be infeasible in order to yield a Pareto optimal solution, and (3) the setting of parameters is not intuitively clear when only one solution point is desired.

17.7 Lexicographic Method

With the *lexicographic method*, preferences are imposed by ordering the objectives according to their importance or significance, rather than by assigning weights. The objective functions are arranged in the order of their importance. Then, the following optimization problems are solved one at a time:

minimize $f_i(\mathbf{x})$ subject to

$$f_j(\mathbf{x}) \leq f_j(\mathbf{x}_j^*); j = 1 \text{ to } (i - 1); i > 1; \quad i = 1 \text{ to } k \quad (17.9)$$

Here, i represents a function's position in the preferred sequence, and $f_j(\mathbf{x}_j^*)$ represents the minimum value for the j th objective function, found in the j th 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 local gradient-based optimization engines. 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 global optimization engine with this approach. In any case, theoretically, the solution is always Pareto optimal. Note that this method is classified as a vector multiobjective optimization method because each objective is treated independently.

Advantages of the method are: (1) it is a unique approach to specifying preferences, (2) it does not require that the objective functions be normalized, and (3) it always provides a Pareto optimal solution. *Disadvantages* are: (1) it can require the solution of many single-objective problems to obtain just one solution point, (2) it needs additional constraints to be imposed, and (3) it is most effective when used with a global optimization engine, which can be expensive.

17.8 Bounded Objective Function Method

The *bounded objective function method* minimizes the single, most important objective function $f_s(\mathbf{x})$ with other objective functions treated as constraints: $l_i \leq f_i(\mathbf{x}) \leq \varepsilon_i$; $i = 1$ to k ; $i \neq s$. l_i and ε_i are the lower and upper bounds for the objective function $f_i(\mathbf{x})$, respectively. In this way, the user imposes preferences by setting limits on the objectives. l_i is obsolete unless the intent is to achieve a goal or fall within a range of values for $f_i(\mathbf{x})$.

The ε -constraint approach (also called the ε -constraint or *trade-off* approach) is a variation of the bounded objective function method in which l_i is excluded. In this case, a systematic variation of ε_i yields a set of Pareto optimal solutions. However, improper selection of the ε -vector can result in a formulation with no feasible solution. Guidelines for selecting

ε -values that reflect preferences are discussed in the literature (Cohon, 1978; Stadler, 1988). A general mathematical guideline for selecting ε_i is provided as follows (Carmichael, 1980):

$$f_i(\mathbf{x}_i^*) \leq \varepsilon_i \leq f_s(\mathbf{x}_i^*) \quad (17.10)$$

A solution to the ε -constraint formulation, if it exists, is weakly Pareto optimal. If the solution is unique, then it is Pareto optimal. Of course, uniqueness can be difficult to verify, although if the problem is convex and if $f_s(\mathbf{x})$ is strictly convex, then the solution is necessarily unique. Solutions with active ε -constraints (and nonzero Lagrange multipliers) are necessarily Pareto optimal (Carmichael, 1980).

Advantages of the method are: (1) it focuses on a single objective with limits on others, (2) it always provides a weakly Pareto optimal point, assuming the formulation gives a solution, (3) it is not necessary to normalize the objective functions, and (4) it gives Pareto optimal solution if one exists and is unique. The only *disadvantage* is that the optimization problem may be infeasible if the bounds on the objective functions are not appropriate.

17.9 Goal Programming

With *goal programming*, goals b_j are specified for each objective function $f_j(\mathbf{x})$. Then, the total deviation from the goals $\sum_{j=1}^k |d_j|$ is minimized, where d_j is the deviation from the goal b_j for the j th objective function. To model the absolute values, d_j is split into positive and negative parts such that $d_j = d_j^+ - d_j^-$ with $d_j^+ \geq 0$, $d_j^- \geq 0$, and $d_j^+ d_j^- = 0$. Consequently, $|d_j| = d_j^+ + d_j^-$ and d_j^+ and d_j^- represent underachievement and overachievement, respectively, where achievement implies that a goal has been reached. The optimization problem is formulated as follows:

$$\text{minimize } \sum_{i=1}^k (d_i^+ + d_i^-) \quad (17.11)$$

$$\text{subject to } f_j(\mathbf{x}) + d_j^+ - d_j^- = b_j; \quad d_j^+, d_j^- \geq 0; \quad d_j^+ d_j^- = 0; \quad i = 1 \text{ to } k$$

In the absence of any other information, goals may be set to the utopia point, i.e., $b_j = f_j^0$. In this case Eq. (17.11) can be considered a type of global criterion method. Lee and Olson (1999) provide an extensive review of applications for goal programming. However, despite its popularity, there is no guarantee that it provides a Pareto optimal solution. Also, Eq. (17.11) has additional variables and nonlinear equality constraints, both of which can be troublesome with larger problems.

Advantages of the method are: (1) it is easy to assess whether the predetermined goal have been reached, and (2) it is easy to tailor the method to a variety of problems. *Disadvantages* are: (1) there is no guarantee that the solution is even weakly Pareto optimal, (2) there is an increased number of variables, and (3) an increase in the number of constraints.

17.10 Selection of Methods

Deciding which multiobjective optimization method is most appropriate or most effective can be difficult. It depends on the nature of the user's preferences and what types of solutions might be acceptable (Floudas *et al.*, 1990). Knowledge about the problem functions can aid in the selection process. Table 17-1 summarizes the following key characteristics of the methods discussed in this chapter and is helpful in selecting the most appropriate method for a particular application:

1. Always provides a Pareto optimal solution.
2. Can provide all the Pareto optimal solutions.
3. Involves weights to express preferences.
4. Depends on the continuity of the problem functions.
5. Uses the utopia point or its approximation.

TABLE 17-1 Characteristics of Multiobjective Optimization Methods

<i>Method</i>	<i>Always yields Pareto optimal point?</i>	<i>Can yield all Pareto optimal points?</i>	<i>Involves weights?</i>	<i>Depends on function continuity?</i>	<i>Uses utopia point?</i>
Genetic	Yes	Yes	No	No	No
Weighted sum	Yes	No	Yes	Problem type and the optimization engine determines this.	Utopia point or its approximation is needed for function normalization, or in the method formulation.
Weighted min-max	Yes ¹	Yes	Yes	Same as above	Same as above
Weighted global criterion	Yes	No	Yes	Same as above	Same as above
Lexicographic	Yes ²	No	No	Same as above	No
Bounded objective fun.	Yes ³	No	No	Same as above	No
Goal programming	No	No	No ⁴	Same as above	No

¹Sometimes the solution is only weakly Pareto optimal.

²Lexicographic method always provides a Pareto optimal solution only if a global optimization engine is used or if the solution point is unique.

³Always weak Pareto optimal if it exists; Pareto optimal if the solution is unique.

⁴Weights may be incorporated into the objective function to represent the relative significance of deviation from a particular goal.

Exercises for Chapter 17*

- 17.1 In the design space, plot the objective function contours for the following unconstrained problem and sketch the Pareto optimal set, which should turn out to be a curve:

minimize

$$f_1 = (x_1 - 0.75)^2 + (x_2 - 2)^2$$

$$f_2 = (x_1 - 2.5)^2 + (x_2 - 1.5)^2$$

Draw the gradients of each function at any point on the Pareto optimal curve. Comment on the relationship between the two gradients.

- 17.2 Sketch the Pareto optimal set for Exercise 17-1 in the criterion space.
- 17.3 In the design space, plot the following constrained problem and sketch the Pareto optimal set:

minimize

$$f_1 = (x_1 - 3)^2 + (x_2 - 7)^2$$

$$f_2 = (x_1 - 9)^2 + (x_2 - 8)^2$$

subject to

$$g_1 = 70 - 4x_2 - 8x_1 \leq 0$$

$$g_2 = -2.5x_2 + 3x_1 \leq 0$$

$$g_3 = -6.8 + x_1 \leq 0$$

17.4 Identify the weakly Pareto optimal points in the plot in Fig. E17-4:

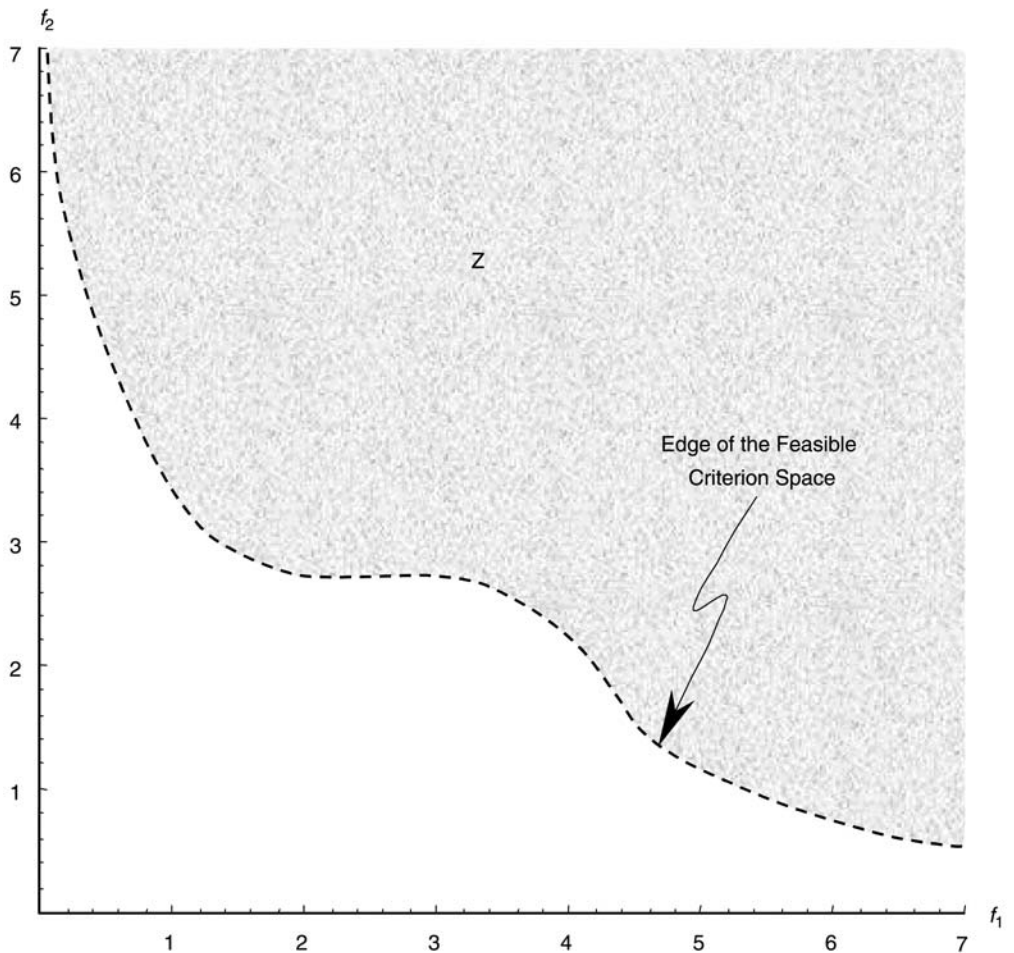


FIGURE E17-4 Identification of weakly Pareto optimal points.

- 17.5 Plot the following global criterion contour in the criterion space, using p -values of 1, 2, 5, and 20 (plot one contour line for each p -value):

$$U = (f_1^p + f_2^p)^{1/p} = 1.0$$

Comment on the difference between the shapes of the different contours. Which case represents the weighted sum utility function (with all weights equal to 1)?

- 17.6 Plot contours for the following min-max utility function in the criterion space:

$$U = \max[f_1, f_2]$$

Compare the shape of these contours with those determined in Exercise 17.5.

- 17.7 Solve the following problem using the KKT optimality conditions of Chapter 4 with the weighted sum method:

minimize

$$f_1 = (x_1 - 3)^2 + (x_2 - 7)^2$$

$$f_2 = (x_1 - 9)^2 + (x_2 - 8)^2$$

Write your solution for the design variables in terms of the two weights w_1 and w_2 . Comment on how the different weights affect the final solution.

- 17.8 Solve the following problem using KKT optimality conditions of Chapter 4 with the weighted sum method:

$$\text{minimize } f_1 = 20(x_1 - 0.75)^2 + (2x_2 - 2)^2$$

$$f_2 = 5(x_1 - 1.6)^2 + 2x_2$$

subject to

$$g_1 = -x_2 \leq 0$$

First, use $w_1 = 0.1$ and $w_2 = 0.9$. Then, resolve the problem using $w_1 = 0.9$ and $w_2 = 0.1$. Comment on the constraint activity in each case.

- 17.9 Formulate the following problem (Stadler and Dauer, 1992) and solve it using Excel with the weighted sum method:

Determine the optimal height and radius of a closed-end cylinder necessary to simultaneously maximize the volume and minimize the surface area. Assume that the cylinder has negligible thickness. The height must be at least 0.1 m, and the radius must be at least half the height. Neither the height nor the radius should be greater than 2.0 m.

Use a starting point of $\mathbf{x}^{(0)} = (1, 1)$. Use the following vectors of weights and comment on the solution that each yields: $\mathbf{w} = (1, 0)$; $\mathbf{w} = (0.75, 0.25)$; $\mathbf{w} = (0.5, 0.5)$; $\mathbf{w} = (0.25, 0.75)$; $\mathbf{w} = (0, 1)$.

- 17.10 Solve Exercise 17.9 using Excel with a weighted global criterion. Use $\mathbf{x}^{(0)} = (1, 1)$, $\mathbf{w} = (0.5, 0.5)$, and $p = 2.0$. Compare the solution with those determined in Exercise 17.9.

17.11 Plot the objective functions contours for the following problem (on the same graph) and solve the problem using the lexicographic method:

minimize

$$f_1 = (x-1)^2(x-4)^2$$

$$f_2 = 4(x-2)^2$$

$$f_3 = 8(x-3)^2$$

Indicate the final solution point on the graph. Assume the functions are prioritized in the following order: f_1, f_2, f_3 , with f_1 being the most important.

18 Global Optimization Concepts and Methods for Optimum Design

Upon completion of this chapter, you will be able to:

- Explain basic concepts associated with finding a global solution for a design problem
- Explain basic ideas, procedures, and limitations of *deterministic* and *stochastic methods* for global optimization
- Use an appropriate method for solving your global optimization problem

The standard design optimization model treated in this text is minimize $f(\mathbf{x})$ with \mathbf{x} in the feasible set S defined as

$$S = \{\mathbf{x} | h_i(\mathbf{x}) = 0, i = 1 \text{ to } p; \quad g_j(\mathbf{x}) \leq 0, j = 1 \text{ to } m\} \quad (18.1)$$

The discrete variables in the problem are treated as explained in Chapter 15. Thus far in this text, we have addressed mainly the problem of finding a local minimum for the cost function in the feasible set. In this chapter, we focus on presentation and discussion of concepts and methods for the global optimum solutions because in some practical applications, it is important to find such solutions as opposed to the local ones. The material for the chapter is introductory in nature and is derived from the work of the author and his coworkers (Arora *et al.*, 1995; Elwakeil and Arora, 1996a,b). Numerous other references are cited in these articles that contain more exposition on the subject (Dixon and Szego, 1978; Evtushenko, 1985; Pardalos and Rosen, 1987; Rinnooy and Timmer, 1987a,b; Törn and Žilinskas, 1989; Pardalos *et al.*, 2002).

18.1 Basic Concepts of Solution Methods

18.1.1 Basic Concepts

Most of the methods presented in this chapter assume continuous variables and functions. For discrete and nondifferentiable problems, the simulated annealing and genetic algorithms are appropriate for global optimization and may be used as described earlier in Chapters 15 and 16. It is also important to note that many methods for global optimization presented in

the literature considered the unconstrained problem only. *It is assumed that the constraints can be treated implicitly using the penalty or augmented Lagrangian type methods that are discussed in Chapter 9.* A possible disadvantage of that approach is that such methods can terminate at infeasible points. Many methods, however, can treat or may even require explicit bound constraints on the design variables. To discuss such methods, let us define a set S_b of feasible points with respect to the explicit bound constraints as

$$S_b = \{x_i \mid x_{il} \leq x_i \leq x_{iu}; \quad i = 1 \text{ to } n\} \quad (18.2)$$

Recall that n is the number of design variables, and x_{il} and x_{iu} are the lower and upper bounds on the i th variable. Note that the feasible set S of Eq. (18.1) is a subset of the set S_b .

It is also noted that *many global optimization methods repeatedly search for local minima in their algorithm.* These methods are relatively easy to implement and use for solving global optimization problems. *It is important to use robust and efficient software to search for local minima.* We shall assume that such an optimization engine is available for use with these global optimization methods.

Before describing the methods for finding global minima, let us first recall definitions of local and global minima presented and discussed in Chapter 4. A point \mathbf{x}^* is called a *local minimum* for the problem $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all \mathbf{x} in a *small feasible neighborhood* of the point \mathbf{x}^* . A point \mathbf{x}_G^* is defined as a *global minimum* for the problem if $f(\mathbf{x}_G^*) \leq f(\mathbf{x})$ for all \mathbf{x} in the feasible set S . A problem can have multiple global minimum points that must have the same cost function value. If the feasible set S is closed and bounded and the cost function is continuous on it, the Weierstrass Theorem 4.1 in Chapter 4 guarantees *existence of a global minimum* point, though finding it is altogether a different matter. At a local optimum point, the Karush-Kuhn-Tucker (KKT) necessary conditions apply (as described in Chapters 4 and 5). Although a global minimum point must also be a local minimum point, *there are no mathematical conditions that characterize a global minimum point*, except when the problem can be shown to be *convex*. However in most practical applications, it is difficult to check for convexity of the problem. Therefore, the problem is generally assumed to be *nonconvex*.

An important question then is: *How do you know that a numerical search process has terminated at a global minimum point? The answer is that, in general, you do not know.* Because of this, it is difficult to define a precise stopping criterion for a computational algorithm for global optimization. Usually, the best solution obtained by an algorithm after it is *allowed to run for a long time* is accepted as the global solution for the problem. In general, the *quality of the solution* depends on how long the algorithm is allowed to run. It is important to note that *the computational effort to solve a global optimization problem is substantial, and it increases enormously as the number of design variables increase.* Thus, *solving global optimization problems remains a challenge from a mathematical as well as a computational view point.* It is noted, however, that some algorithms can be implemented on parallel processors which can reduce the “wall clock” time to solve the problem.

It is seen that because of the lack of global optimality conditions for general problems, a global solution to the problem can be obtained only by an *exhaustive search of the design space* (the feasible set S). The procedure for such a search is to specify some sample points in the set S_b and evaluate the cost function at those points. The point where the function has the smallest value is taken as the global minimum point. It is seen that the location and value of the global minimum depend on the sample size. An exact solution for the problem requires an infinite number of calculations. Generally, this infinite calculation is avoided by accepting the best solution as a global minimum point obtained by an algorithm after it is allowed to run for a sufficiently long time. When a point within a distance ε from \mathbf{x}_G^* is sought, *many strategies* exist that only require a *finite number* of function evaluations. These strategies, however, are of limited practical use since ε cannot be specified because \mathbf{x}_G^* is not known.

Thus, either a *further restriction* on the class of cost functions or a *further relaxation* of what is required of an algorithm is necessary.

18.1.2 Overview of Methods

The global optimization methods can be divided into two major categories: *deterministic* and *stochastic*. This classification is mainly based on whether the method incorporates any stochastic elements to solve the global optimization problem. Deterministic methods find the global minimum by an exhaustive search over the set S_b . The success of the method can be guaranteed for only the functions that satisfy certain conditions. We shall describe four deterministic methods: covering, zooming, generalized descent, and tunneling.

Several stochastic methods have been developed as some variation of the pure random search. Some methods are useful for only discrete optimization problems while others can be used for both discrete and continuous problems. All the stochastic methods involve random elements to determine the global minimum point, each one trying to reduce the computational burden of pure random search. At the outset, a random sample of points in the set S_b is picked. Then each method manipulates the sample points in a different manner. In some cases the two operations are simultaneous; i.e., a random point is picked and manipulated or used before the next one is picked. We shall briefly describe some of the methods such as multistart, clustering, control random search, acceptance-rejection, stochastic integration, stochastic zooming, and domain elimination.

In the remaining sections of this chapter, we describe basic concepts and ideas of various methods for global optimization. Algorithms for some of the methods are described and discussed to give the student a flavor of the type of calculations needed to find a global solution for the design problem. Some of the methods describe calculations for the global minimum of the cost function without reference to constraints. It is assumed in these methods that the constraints are used to define a penalty function, which is then minimized. Some of the algorithms have been implemented on the computer to evaluate their performance on mathematical programming test problems as well as structural design problems. These numerical experiments are described, and performance results of the methods are discussed.

18.2 Overview of Deterministic Methods

Deterministic methods find the global minimum by an exhaustive search over the set S_b . If an absolute guarantee of success is desired for such a method, additional assumptions about the cost function are needed to avoid huge calculations. The most popular approach is to assume the *Lipschitz continuity condition* for the function: There exists a *Lipschitz constant* L such that for all \mathbf{x} , \mathbf{y} in the set S_b , $|f(\mathbf{x}) - f(\mathbf{y})| \leq L\|\mathbf{x} - \mathbf{y}\|$, i.e., *the rate of change of the function is bounded*. The upper bound on the rate of change of $f(\mathbf{x})$ implied by the Lipschitz constant can be used in various ways to perform an exhaustive search over the set S_b (Evtushenko, 1985). Unfortunately, in practice it is hard to verify whether a function satisfies such a condition for all points in the set S_b .

Deterministic methods for global optimization are further classified as *finite exact* and *heuristic methods*. *Finite exact methods* provide an *absolute guarantee* that the global minimum will be found in a *finite* number of steps. Generally, the number of steps is very large, and so the methods require large computational effort, especially when the number of design variables is more than two. However, for some problems, it is essential to find the global minimum with absolute guarantee, irrespective of the computational effort needed. Since no other method gives an absolute guarantee of finding the global minimum in a finite number of steps, these methods become important. *Heuristic methods*, on the other hand, offer only an *empirical guarantee* of finding the global optimum.

18.2.1 Covering Methods

As the name implies, the basic idea of covering methods is to “cover” the set S_b by evaluating the cost function at all the points in this search for the global minimum. This is, of course, an infinite calculation and is therefore impossible to implement and use. Thus all the covering methods devise procedures to evaluate the functions at selected points but still cover the entire set S_b implicitly. Some covering methods take advantage of certain properties of the cost function to define a mesh of points that may or may not be uniform for evaluating functions at these points. Some of the covering methods are relatively efficient but can treat only simple cost functions (which occur only in standard test problems). In these methods, upper and lower bounds on the cost function over a subset of S_b are computed by *interval arithmetic*. Different means to exclude inferior intervals are then used. The *branch-and-bound* methods discussed in Chapter 15 are also based on such ideas. Other methods successively form closer approximations (of given functions) that can be separated into convex and concave terms.

The covering method of Evtushenko (1985) uses a nonuniform mesh to cover the set S_b . In the method, an approximation to the solution point \mathbf{x}_G^* is obtained for a given positive tolerance ε such that it belongs to the set A_ε of points with cost function values less than $f(\mathbf{x}_G^*) + \varepsilon$, i.e., A_ε is defined as

$$A_\varepsilon = \{\mathbf{x} \in S \mid (f(\mathbf{x}) - \varepsilon) \leq f(\mathbf{x}_G^*)\} \quad (18.3)$$

The set A_ε can never be constructed since \mathbf{x}_G^* is not known. The solution, however, is guaranteed to belong to it; i.e., it is within ε of the global minimum value. In some covering methods, the mesh density is determined using the *Lipschitz constant* L . The upper bound on the rate of change of $f(\mathbf{x})$ implied by the Lipschitz constant is used in various ways to sequentially generate a mesh and perform an exhaustive search over the set S_b . Unfortunately, in practice it is hard to verify whether a function satisfies such a condition for all points in the set S_b . Also, the computational effort required to compute L is substantial. Therefore, only an approximation to L can be used.

In Evtushenko’s method, the mesh points are generated as centers of hyperspheres. The union of these spheres has to completely cover S_b for the approximate solution to be valid. The covering is done sequentially; one sphere after another is constructed until the entire set is covered. Therefore, the total number of mesh points is not known until the covering is complete. In multidimensional problems, covering by hyperspheres is difficult and inefficient as the hyperspheres must overlap to cover the entire set S_b . Therefore, hypercubes inscribed in the hyperspheres are used instead. In two dimensions the design space is filled with squares; in three dimensions it is filled with cubes and so on. The resulting mesh is nonuniform in the first variable and uniform in the rest of the variables. Since finding the true value of the Lipschitz constant L is a difficult task, a smaller approximation for L and a larger value for the tolerance ε are used initially. Then the approximation for L is increased and that of ε is decreased and the entire covering procedure is repeated. The repetition is continued until the difference between two consecutive solutions is less than ε . In some methods, departing from purely deterministic procedures, the Lipschitz constant is estimated using some statistical models of the cost function. An advantage of Evtushenko’s method is that it yields a guaranteed estimate of the global minimum for any upper bound approximation of the Lipschitz constant. It is seen that the covering methods are generally not practical for problems having more than two variables. Two variable problems can be solved more efficiently by the graphical optimization method of Chapter 3.

18.2.2 Zooming Method

The *zooming method* was designed especially for problems with general constraints. The method uses a target value for the global minimum of the cost function. Once the target is

achieved, it is reduced further to “zoom-in” on the global minimum. The method combines a local minimization method with successive truncation of the feasible set S to eliminate regions of local minima to *zoom-in* on the global solution. The basic idea is to initiate the search for a constrained local minimum from any point—feasible or infeasible. Once a local minimum point has been found, the problem is redefined in such a way that the current solution is eliminated from any further search by adding the following constraint to the problem:

$$f(\mathbf{x}) \leq \gamma f(\mathbf{x}^*) \quad (18.4)$$

where $f(\mathbf{x}^*)$ is the cost function value at the current minimum point and $0 < \gamma < 1$ if $f(\mathbf{x}^*) > 0$, and $\gamma > 1$ if $f(\mathbf{x}^*) < 0$. The redefined problem is solved again and the process continued until no more minimum points can be found. The zooming method appears to be a good alternative to stochastic methods (discussed later in this chapter) and other methods for constrained global optimization problems. It is quite simple to use: the formulation is modified slightly by adding the zooming constraint of Eq. (18.4), and an existing local minimization routine is used. However, there are certain *limitations* of the method. As the target level for the global minimum of the cost function is lowered, the feasible set for the problem keeps on shrinking. It may also result in a disjointed feasible set. Therefore as the global minimum is approached, finding even a feasible point for the redefined problem becomes quite difficult. Several different trial starting points needs to be tried before declaring the redefined problem to be infeasible and accepting the previous local minimum as the global minimum. The only stopping criterion is the limit on the number of trials allowed to search a feasible point for the reduced feasible set. An improvement of the method is described later where some stochastic elements are introduced into the computational procedure.

18.2.3 Methods of Generalized Descent

Generalized descent methods are classified as *heuristic deterministic methods*. These methods are a generalization of the descent methods described earlier in Chapters 8 and 9. In those methods, finite descent steps are taken along straight lines, i.e., the search directions. For nonquadratic problems, it is sometimes difficult to find a suitable step size along the search direction. Therefore, it may be more effective if we deliberately follow a *curvilinear path* in the design space (also called *trajectories*). Before describing the generalized descent methods for global optimization, we shall describe the basic ideas of *trajectory methods* that generate curvilinear paths in the design space in search of minimum points.

A trajectory can be considered as a design history of the cost function from the starting point $\mathbf{x}^{(0)}$ to a local minimum point \mathbf{x}^* . Let the design vector \mathbf{x} be dependent on the parameter t that monotonically increases along the solution curve $\mathbf{x}(t)$ and is zero at $\mathbf{x}^{(0)}$. The simplest path from an arbitrary initial point $\mathbf{x}^{(0)}$ to \mathbf{x}^* is a continuous steepest descent trajectory given as solution of the vector differential equation:

$$\dot{\mathbf{x}}(t) = -\nabla f(\mathbf{x}), \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}^{(0)} \quad (18.5)$$

where an “over dot” represents the derivative with respect to t . We can also use a continuous Newton’s trajectory by changing the right side of Eq. (18.5) to $-\mathbf{H}(\mathbf{x})^{-1}\nabla f(\mathbf{x})$, where $\mathbf{H}(\mathbf{x})$ is the Hessian of the cost function that is assumed to be nonsingular for all \mathbf{x} . It is noted that good software is available to solve the first-order differential equation (18.5).

The *generalized descent methods* for global optimization are extensions of the foregoing trajectory methods. In these methods, the trajectories are solutions of certain second-order differential equations rather than the first order equation (18.5). The search for the global minimum is based on solution properties of these differential equations. The most important property is that their trajectories pass through the majority of the stationary points of the cost

function (or in their neighborhood). Certain conditions determine if the trajectory will pass through all the local minima of the function. In that case, the global minimum is guaranteed to be found. The differential equations use the function values and function gradients along the trajectories.

There are two types of generalized descent methods: (1) the *trajectory methods* that modify the differential equation describing the local descent trajectory in such a way to make it converge to a global rather than a local minimum, and (2) the *penalty methods* that apply the standard local algorithm repeatedly to a modified cost function so as to prevent the descent trajectory from converging to local minima that were previously found. Examples of penalty methods are: algebraic functions, filled function, and tunneling. In this section we shall describe the trajectory methods only. The tunneling methods in the class of penalty methods shall be described in the next section.

Alternation of Descents and Ascents Trajectory methods have been implemented in two ways: alternation of descents and ascents and the so-called golf methods. The first method consists of three subalgorithms that are modifications of the local descent algorithms. These are: *descent* to a local minimum, *ascent* from a local minimum, and *pass through* a saddle point. First, to descend to a local minimum from a starting point, we use a combination of steepest descent and Newton's methods based on whether the Hessian matrix of the cost function is positive definite or not at a point along the trajectory (this is the *modified Newton's method* for local minimization). Second, to get from the local minimum point to a saddle point, we use the eigenvector corresponding to the maximum eigenvalue of the Hessian as the search direction. Third, to pass through the saddle point, we use Newton's method. To descend to the next local minimum, we use the direction of the last step of the passing operation as a starting direction for the descent operation. The three operations are repeated until some stopping criterion is satisfied. Note that all local minimum points are recorded so that if the trajectory retraces itself, a new initial point is chosen to restart the algorithm. Disadvantages of this method are the large number of function evaluations wasted in ascending from a local minimum, and difficulties with solving problems of dimension larger than two. It is also difficult to apply the method if we do not have an expression for the cost function gradient.

Golf Methods The golf methods make analogy with mechanics of inertial motion of a particle of mass m moving in a force field. The resulting trajectory is analogous to that of the optimization problem. Mathematically, the assignment of mass means introducing a second-order term into the particle's equation of motion. Taking the mass as a function of time $m(t)$, the particle is thus moving in a force field defined by the cost function $f(\mathbf{x})$ and subjected to a dissipating or nonconservative resistance force (e.g., air resistance force) given by $-n(t)\dot{\mathbf{x}}(t)$, where $n(t)$ is the resistance function. The force field of $f(\mathbf{x})$ is given as $-\nabla f(\mathbf{x})$. Thus, the motion of the particle is described by the system of differential equation:

$$m(t)\ddot{\mathbf{x}}(t) - n(t)\dot{\mathbf{x}}(t) = -\nabla f(\mathbf{x}), \quad m(t) \geq 0 \quad n(t) \leq 0 \quad (18.6)$$

where $\dot{\mathbf{x}}(t)$ and $\ddot{\mathbf{x}}(t)$ are the velocity and acceleration vectors of the particle, respectively. Under some conditions, the trajectory, which is a solution of the system of equations, converges to a local minimum point of $f(\mathbf{x})$. Moreover, the trajectory leaves some local minima that are not deep enough, and hence the name, golf methods. It is obvious that algorithm efficiency is a function of the mass and resistance functions $m(t)$ and $n(t)$. For some functions, the differential equation is simplified by assuming the mass of the particle as 1 and a frictionless force field [i.e., $m(t) = 1$ and $n(t) = 0$]. In this case the differential equation gets

simplified to $\ddot{\mathbf{x}}(t) = -\nabla f(\mathbf{x})$. Such a class of functions is encountered if $f(\mathbf{x})$ is an interpolation of noisy data from some experiments.

18.2.4 Tunneling Method

The *tunneling method* falls into the class of heuristic generalized descent penalty methods. The method was initially developed for unconstrained problems and then extended for constrained problems (Levy and Gomez, 1985). The basic idea is to execute the following two phases successively until some stopping criterion is satisfied: the *local minimization* phase and the *tunneling* phase. The first phase consists of finding a local minimum \mathbf{x}^* for the problem using any reliable and efficient method. The tunneling phase determines a starting point that is different from \mathbf{x}^* but has a cost function value smaller than or equal to the known minimum value. However, finding a suitable point in the tunneling phase is also a global problem that is as hard as the original problem. When a rough estimate of the global minimum is required, use of the tunneling or zooming methods is justified instead of a method that can guarantee a global minimum at the expense of a large computational effort.

The basic idea of the tunneling method is depicted graphically in Fig. 18-1 for the case of the one-dimensional problem. Starting from the initial minimum point $\mathbf{x}^{*(1)}$, the method *tunnels* under many other local minima, and locates a new starting point $\mathbf{x}^{0(2)}$. From there, a new local minimum is found, and the process is repeated. The tunneling phase is accomplished by finding a root of the nonlinear *tunneling function*, $T(\mathbf{x})$. This function is defined in such a way that it avoids previously determined local minima and the starting points. The new point found during the tunneling phase should not be a local minimum either because the local minimization procedure cannot be started from such a point. Therefore, if the tunneling phase yields a local minimum point, a new root of the tunneling function is sought. Once a suitable point is obtained through the tunneling phase, local minimization is started to obtain a new local minimum point. The two phases are repeated until no suitable roots of the tunneling function can be found, which is realized numerically when $T(\mathbf{x}) \geq 0$ for all \mathbf{x} . We note here that such a criterion is very expensive to satisfy in terms of the number of function evaluations needed. The first few tunneling phases are relatively efficient, i.e., they

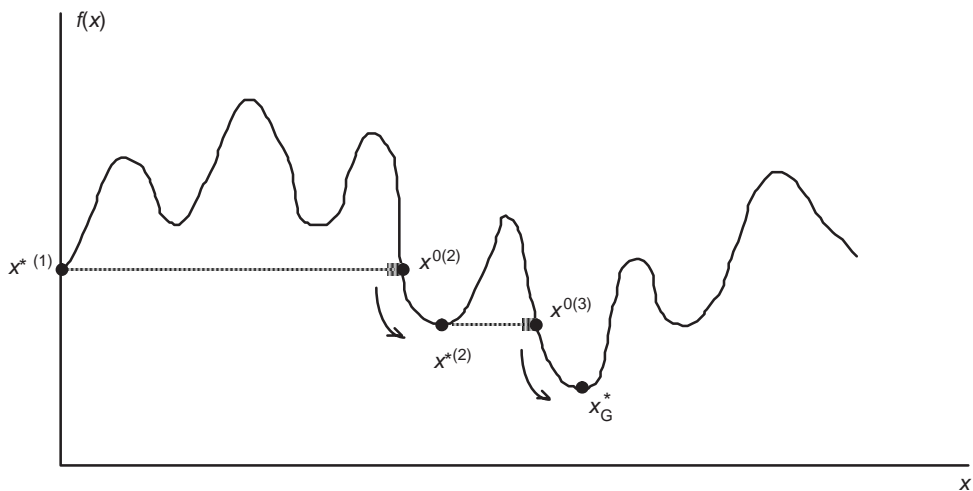


FIGURE 18-1 Basic concept of tunneling method. The method tunnels below irrelevant minima to approach the global minimum.

require only little computing effort. As the tunneling level approaches the global minimum, the number of computations increases because there are fewer roots of the tunneling function. This difficulty is similar to the one noted for the zooming method.

The tunneling method has the *global descent property*: the local minima obtained by the minimization phase approach the global minimum in an orderly fashion. Tunneling takes place below irrelevant local minima regardless of their number or location. Because of this property, the tunneling method can be efficient relative to other methods, especially for problems with a large number of local minima. The method has the advantage that a point with a smaller cost function value is reached at each iteration. Therefore a point with a relatively smaller cost function value is obtained very quickly, as with the zooming method. Such a solution is acceptable for some engineering applications. In such cases, use of the tunneling or the zooming method is justified instead of a method that can guarantee a global minimum at the expense of a large computational effort.

18.3 Overview of Stochastic Methods

Let S_* be the *set of all the local minima* of the optimization problem. The aim of many stochastic methods is to determine this set S_* . Then, the best point of the set is claimed as the global minimum point. Generally, far better results have been obtained using *stochastic* methods than deterministic methods. Stochastic methods usually have *two phases*: a global phase and a local phase. In the *global phase*, the function is evaluated at a number of randomly sampled points. In the *local phase*, the sample points are manipulated, e.g., by means of local searches, to yield candidate global minima. The global phase is necessary because there is no local improvement strategy that, starting from an arbitrary point, can be guaranteed to converge to the global minimum. The global phase locates a candidate global minimum point in every subset of the feasible set S_b to ensure *reliability* of the method. Local search techniques are efficient tools for finding a point with a relatively small function value. Therefore, the local phase is incorporated into the stochastic methods to improve their *efficiency*. A challenge for global optimization algorithms is to increase their efficiency while maintaining reliability. There are many stochastic methods for global optimization, such as random search, multistart, clustering, controlled random search, simulated annealing, acceptance-rejection, stochastic integration, genetic, and tabu search. We shall describe only the underlying ideas that lay the foundation for computations of the methods. More details can be found in Arora and coworkers (1995) and other references cited therein.

Most of the *stochastic methods* are based on some variation of the pure random search. The stochastic ideas are used in two ways: (1) to develop stopping criteria and (2) to develop techniques to approximate the *region of attraction* for a local minimum point, which is defined as follows: *When the search for the local minimum started from a point within a certain region around the minimum converges to the same minimum point, the region is called the region of attraction for that local minimum.* The goal of many stochastic methods is to develop good approximations for the regions of attraction for local minima so that the search for a local minimum is performed only once.

Usually, most of the stochastic algorithms use *uniform distribution* of sampling over the set S_b . However, mechanisms for modifying the sampling distribution based on the information obtained in previous iterations may be more appropriate. A stochastic approximation of the type used in sampling can be used to determine a sampling distribution that would peak in unexplored regions of attraction to discover new local minima. Even though the stochastic methods do not offer an *absolute guarantee* of success, *the probability that a point within a distance ε of \mathbf{x}_G^* is found approaches 1 as the sample size increases.*

Note that since some stochastic methods use random processes, such as simulated annealing and genetic algorithms, an algorithm run at different times can generate different design histories and local minima. Therefore, a particular problem needs to be run several times before the solution is accepted as the global optimum.

18.3.1 Pure Random Search

Pure random search is the *simplest stochastic method* for global optimization. Most other stochastic methods are some variation of the pure random search. Though very inefficient, it is described here to introduce a basis for many other stochastic methods. The method consists only of a global phase: evaluate $f(\mathbf{x})$ at N sample points drawn from a random uniform distribution over the set S_b . The smallest function value found is the candidate global minimum for $f(\mathbf{x})$. The pure random search is asymptotically guaranteed to converge, in a probabilistic sense, to the global minimum point. The method is quite inefficient because of the large number of function evaluations required to provide such a guarantee. A simple extension of the method is the so-called *single start* method. In this, a single local search is performed (if the problem is continuous) starting from the best point in the sample set at the end of the pure random search.

18.3.2 Multistart Method

The multistart method is one of several extensions of the pure random search where a local phase is added to the global phase to improve efficiency. In multistart, in contrast to the single start method, each sample point is used as a starting point for the local minimization procedure. The best local minimum point found is a *candidate* for the global minimum \mathbf{x}_G^* . The method is reliable, but it is not efficient since many sample points will lead to the same local minimum. The algorithm consists of three simple steps: (1) take a random point $\mathbf{x}^{(0)}$ from a *uniform distribution* over the set S_b , (2) start a local minimization procedure from $\mathbf{x}^{(0)}$, and (3) return to Step 1 unless a *stopping criterion* is satisfied. Once the stopping criterion is satisfied, the local minimum with the smallest function value is taken as the global minimum \mathbf{x}_G^* . It can be seen that a particular local minimum may be *reached several times* starting from different points. Therefore, strategies to eliminate this inefficiency in the algorithm have been developed; they are discussed in the following sections.

Stopping Criterion Several ideas about terminating an algorithm have been proposed; however, most of them are not practical. Here we describe a criterion that has been used most often. Since the starting points of the multistart method are uniformly distributed over S_b , a local minimum has a fixed *probability* of being found in each trial. In a *Bayesian approach*, in which the unknowns are themselves assumed to be random variables with a *uniform prior distribution*, the following result can be proved: Given that M distinct local minima have been found in L searches, the *optimal Bayesian estimate* of the unknown number of local minima K is given by

$$K = \text{integer} \left[\frac{M(L-1)}{L-M-2} \right] \quad \text{provided } L \geq M - 3 \quad (18.7)$$

The multistart method can be stopped when $M = K$. It can be shown that this stopping rule can be used for other methods as well.

18.3.3 Clustering Methods

Clustering methods remove the inefficiency of the multistart method by trying to use the local search procedure only once for each local minimum point. To do this, random sample points are linked into groups to form clusters. Each cluster is considered to represent one region of

attraction for a local minimum point. Each local minimum point has a *region of attraction* such that a search initiated from any point in the region converges to the same local minimum point. Four clustering methods have been used for development of the regions of attraction: density clustering, single linkage, mode analysis, and vector quantization multistart. They differ in the way in which these regions of attraction are constructed. A major disadvantage of the clustering methods is that their performance depends heavily on the dimension of the problem, i.e., the number of design variables.

Let A_N be a set of random points drawn from a uniform distribution over the set S_b (details of how to define A_N are given later). In the clustering methods, a preprocessing of the sample is performed to produce regions that are likely to contain local minima. This can be done in two ways: reduction and concentration. In *reduction*, a set A_q of sample points having the cost function values smaller than or equal to some value f_q is constructed, i.e.,

$$A_q = \{\mathbf{x} \in A_N \mid f(\mathbf{x}) \leq f_q\} \quad (18.8)$$

This is called an f_q -*level set* of $f(\mathbf{x})$ or simply the reduced set, and points \mathbf{x} in the set A_q are called the *reduced sample points*. The set A_q may be composed of a number of components that are disjointed. Each of the components will contain at least one local minimum point. Figure 18-2(A) shows an example of a uniform sample in a set, and Fig. 18-2(B) shows the reduced sample points—the set A_q . The set consists of three components each containing one local minimum point. A component of A_q is called a cluster, which is taken as an approximation to the region of attraction. Note that depending on the value of f_q , a component may contain more than one local minimum point. Furthermore, the local minimum points \mathbf{x}^* having function values higher than f_q will not belong to A_q and therefore may not be found.

In the second preprocessing procedure, called *concentration*, a few *steepest descent* steps are applied to every sample point. However in this case, unlike in reduction, the transformed points are not uniformly distributed. Usually, in clustering methods, a uniform distribution is assumed; therefore, the former method of transformation is preferred.

Four clustering methods are available in the literature: *density clustering*, *single linkage clustering*, *mode analysis clustering*, and *vector quantization multistart*. In these methods it is assumed that (1) all local minima of $f(\mathbf{x})$ lie in the interior of S_b , (2) stationary points are

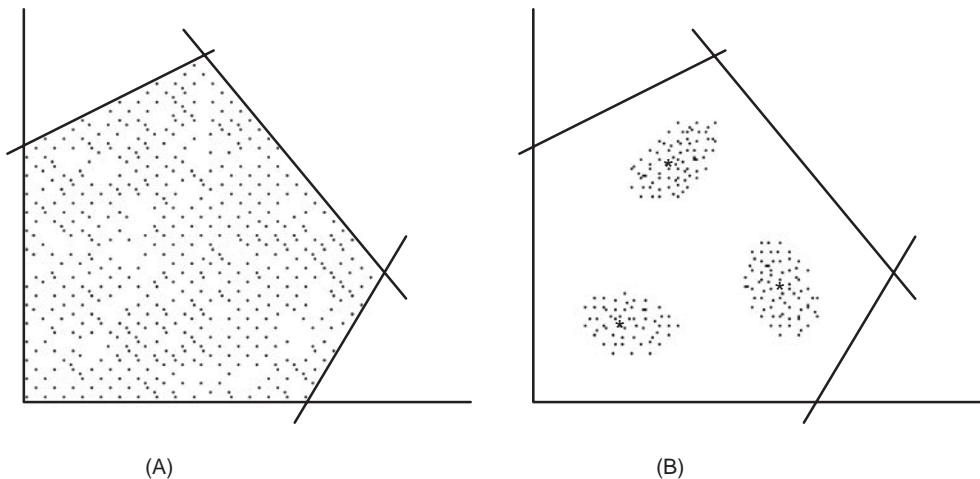


FIGURE 18-2 An example of random and reduced sample points. (A) Random sample of points. (B) Reduced sample points. The * indicates a local minimum point.

isolated, and (3) local search is always that of descent. The methods execute a basic algorithm a number of times. In every iteration, a set of A_N consisting of κN sample points is used from a uniform random distribution, where κ is an integer containing the number of times the algorithm has been executed. That is, the sample size keeps on increasing with each execution of the algorithm. Before clustering, the sample is reduced to produce the set A_q defined in Eq. (18.8). The clustering algorithm is then applied to A_q . The iterations are continued until a stopping criterion is satisfied. The stopping criterion used for multistart can be also applied to all the clustering methods. In applying these rules, we have to assume that the way A_q changes with different samples does not affect analysis. More importantly, we also have to assume that each local minimum with a function value smaller than f_q is actually found.

In *density clustering*, clusters are identified based on the density of the reduced sample points defined in Eq. (18.8); hence the name “density clustering.” Regions of attraction are approximated by hyperspheres or ellipsoids with centers at local minimum points. Reduced sample points are added to clusters based on their distance from the centers (also called the *seed points*), i.e., all points within a critical radius of a center belong to the cluster. A cluster is started with a local minimum point as a seed and then expanded in stages by increasing its critical radius. All points within the new radius are added to the cluster and so on. At the end of a stage, the best unclustered point is used in a local search procedure to find a local minimum point. If the local minimum found is new, then it is taken as the seed for a new cluster; otherwise that minimum point is the seed for an already existing cluster that needs to be expanded. This is continued until all the reduced sample points are clustered.

In *single linkage clustering*, a better approximation to the clusters is achieved by not enforcing a particular shape. Points are linked to others in their proximity as opposed to linkage to the clusters’ centers or seeds. A point is assigned to a cluster if it falls within a critical distance r_κ from any point that already belongs to that cluster.

The density clustering and the single linkage clustering methods use information at only two points at a time. In the *mode analysis method*, on the other hand, clusters are formed by using more information. In this, the set S_b is partitioned into nonoverlapping, small hypercubic *cells* that cover S_b entirely. The cell is said to be *full* if it contains at least G reduced sample points, otherwise it is empty.

In the *vector quantization method*, *theory of lattices* and *vector quantization* are used to form clusters. The basic idea is to cluster cells rather than sample points, like mode analysis. In this, the entire space S_b is divided into a finite number of cells and a *code point* is associated with each cell. The code point is then used to represent all the points in that cell during the clustering process. The point with the smallest function value of a cell is the most suitable code point. Further, code points need not be sample points; they can be generated independently. They may also be centroids of the cells. Identification of a cluster is done using *vector quantization* of the reduced sample points. The aim is to approximate the clusters in a more efficient way than the previous three methods.

18.3.4 Controlled Random Search

The *basic idea of the controlled random search* (CRS) method, which is another variation of the pure random search, is to use the sample points in such a way so as to move toward the global minimum point (Price, 1987). The method does not use gradients of the cost function and so continuity and differentiability of the functions are not required. It uses the idea of a *simplex*, which is a geometric figure formed by a set of $n + 1$ points in the n -dimensional space (recall that n is the number of design variables). When the points are equidistant, the simplex is said to be *regular*. In two dimensions, the simplex is just a triangle; in three dimensions, it is a tetrahedron (see Fig. 18-3), and so on. The method has global and local phases. The notation used in the *global phase* of the algorithm that follows, is defined as

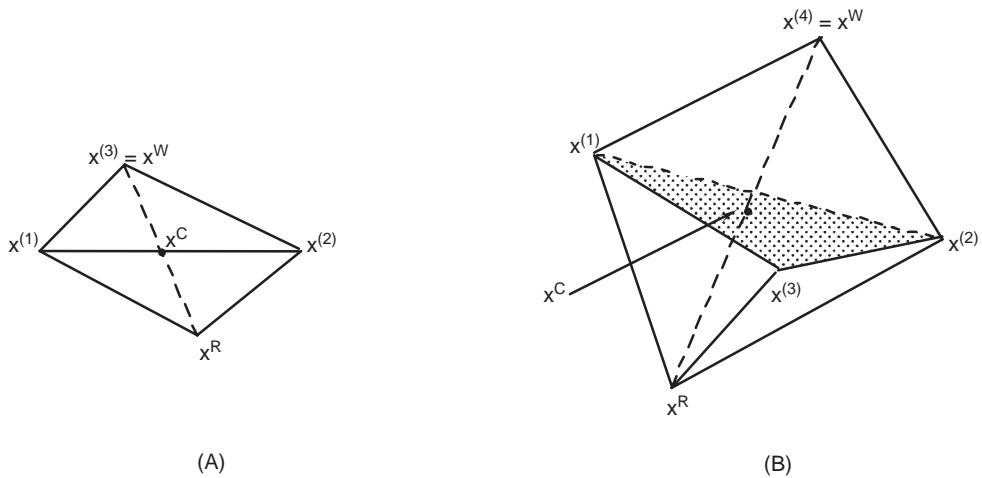


FIGURE 18-3 Reflection operations. (A) Two-dimensional simplex. (B) Three-dimensional simplex. The points \mathbf{x}^W , \mathbf{x}^C , and \mathbf{x}^R lie on a straight line.

- \mathbf{x}^W, f^W : worst point and the corresponding cost function value (largest)
- \mathbf{x}^L, f^L : best point and the corresponding cost function value (smallest)
- \mathbf{x}^C, f^C : centroid of n points and the corresponding cost function value
- \mathbf{x}^P, f^P : trial point and the corresponding cost function value

Step 1. Generate N random points uniformly distributed over S_b . Evaluate the cost function at the points.

Step 2. Find the worst point \mathbf{x}^W with function value f^W (largest) and the best point \mathbf{x}^L with function value f^L (smallest).

Step 3. Let $\mathbf{x}^{(1)} = \mathbf{x}^L$. Randomly choose n distinct points $\mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n+1)}$ from the remaining $N - 1$ sample points. Determine the centroid \mathbf{x}^C of the points $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}$. Compute a new trial point $\mathbf{x}^P = 2\mathbf{x}^C - \mathbf{x}^{(n+1)}$.

Step 4. If \mathbf{x}^P is feasible, evaluate f^P , and go to Step 5. Otherwise, go to Step 3.

Step 5. If $f^P < f^W$, then replace \mathbf{x}^W by \mathbf{x}^P and go to Step 6. Otherwise, go to Step 3.

Step 6. If a stopping criterion is satisfied, then stop. Otherwise, go to Step 2.

As the algorithm proceeds, the current set of n points tends to cluster around the minimum point. Note that the point $\mathbf{x}^{(n+1)}$ used in the calculation of the new trial point \mathbf{x}^P in Step 3 is arbitrarily chosen. This point is called the *vertex* of the simplex. Once the global phase has terminated, the local phase starts. The basic idea of the local phase is to compare cost function values at the $n + 1$ vertices of the simplex and move this simplex gradually toward the optimum point. The movement of the simplex is achieved by using three operations known as *reflection*, *expansion*, and *contraction*. The following additional notation is used in describing these operations:

- \mathbf{x}^S, f^S : the second worst point and the corresponding cost function value
- \mathbf{x}^R, f^R : reflected point and the corresponding cost function value
- \mathbf{x}^E, f^E : expansion point and the corresponding cost function value
- \mathbf{x}^Q, f^Q : contraction point and the corresponding cost function value

Reflection Let $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n+1)}$ be the $n + 1$ points that define the simplex and let the worst point (\mathbf{x}^W) be the vertex with largest cost function value. It can be expected that the point \mathbf{x}^R

obtained by reflecting \mathbf{x}^W in the opposite face of the simplex will have a smaller cost function value. If this is the case, then a new simplex can be constructed by rejecting the point \mathbf{x}^W from the simplex and including the new point \mathbf{x}^R . In Fig. 18-3(B), the original simplex is given by points $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$, and $\mathbf{x}^{(4)} = \mathbf{x}^W$ and the new simplex is given by $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$, and \mathbf{x}^R . The point \mathbf{x}^C is the centroid of the n points of the original simplex excluding \mathbf{x}^W . It is seen that the direction of movement of the simplex is always away from the worst point. Mathematically, the reflection point \mathbf{x}^R is given by

$$\mathbf{x}^R = (1 + \alpha_R)\mathbf{x}^C - \alpha_R\mathbf{x}^W, \quad \text{with} \quad 0 < \alpha_R \leq 1 \quad (18.9)$$

Expansion If the reflection procedure produces a better point, one can generally expect to reduce the function value further by moving along the direction \mathbf{x}^C to \mathbf{x}^R . An expansion point \mathbf{x}^E along this direction is calculated as (its use is explained later the local phase algorithm):

$$\mathbf{x}^E = (1 + \alpha_E)\mathbf{x}^C - \alpha_E\mathbf{x}^W, \quad \text{with} \quad \alpha_E > 1 \quad (18.10)$$

Contraction If the point obtained by reflection is not satisfactory, a contraction point \mathbf{x}^Q along the direction \mathbf{x}^C to \mathbf{x}^R can be calculated as (its use is explained later in the local phase algorithm):

$$\mathbf{x}^Q = (1 + \alpha_Q)\mathbf{x}^C - \alpha_Q\mathbf{x}^W, \quad \text{with} \quad -1 < \alpha_Q < 0 \quad (18.11)$$

The *local search algorithm* given below uses the relations in Eqs. (18.9) to (18.11) with $\alpha_R = 1$, $\alpha_E = 3$, and $\alpha_Q = -1/2$, respectively. The $n + 1$ best points of the random sample of the global phase constitute a simplex in the n -dimensional space for this algorithm. The algorithm does not use the gradient of the cost function $f(\mathbf{x})$. Therefore, a compatible local search procedure that also does not use gradients is needed. An algorithm that is an adaptation of basic *Nelder-Mead simplex algorithm* (1965) for general constrained optimization problems may be used.

- Step 1.* For the simplex formed of $n + 1$ points, let \mathbf{x}^W be the worst point and \mathbf{x}^C be the centroid of the other n points. Let \mathbf{x}^S be the second worst point of the simplex with function value f^S . Compute three trial points based on reflection, expansion, and contraction as $\mathbf{x}^R = 2\mathbf{x}^C - \mathbf{x}^W$, $\mathbf{x}^E = 4\mathbf{x}^C - 3\mathbf{x}^W$, $\mathbf{x}^Q = (\mathbf{x}^C + \mathbf{x}^W)/2$.
- Step 2.* If \mathbf{x}^R is not in S_b , then go to Step 4. Otherwise, evaluate the function value f^R at \mathbf{x}^R . If $f^R < f^S$, then go to Step 3. Otherwise, go to Step 4.
- Step 3. Expansion:* If \mathbf{x}^E is not in S_b , then accept \mathbf{x}^R as the replacement point and go to Step 5. Otherwise calculate the function value f^E at \mathbf{x}^E . If $f^E < f^S$, then accept \mathbf{x}^E as the replacement point and go to Step 5. Otherwise, accept \mathbf{x}^R as the replacement point and go to Step 5.
- Step 4.* If \mathbf{x}^Q is infeasible, then stop; no further improvement is possible. Otherwise evaluate the function value f^Q at \mathbf{x}^Q . If $f^Q < f^S$, then accept \mathbf{x}^Q as the replacement point and go to Step 5. Otherwise, stop.
- Step 5.* Update the simplex by replacing \mathbf{x}^W with the replacement point. Return to Step 1.

The global and local phases of the method described in the foregoing are combined as follows: Execute the global phase and generate a new trial point \mathbf{x}^P in Step 3. Let the N sample points be sorted in the descending order of their cost function values. If \mathbf{x}^P is feasible and falls within the bottom $n + 1$ points of the sample, then execute the local phase starting with

those $n + 1$ points as a simplex. Continue execution of the two phases until the global phase stops in Step 6.

The following features of the method should be noted. The local phase operates only on the best $n + 1$ points in the database of sample points. Thus, it has minimal effect on the performance of the global phase. In Step 2 of the composite algorithm, the local phase may improve the best point in the database. Thus, it tends to speed up the convergence as the global phase always uses the best point. However, this may reduce, to a small degree, the global search capability. If desired, it is easy to counter this effect by requiring the global phase to not include the best point in some iterations (i.e., in Step 3 of the algorithm, choose all the $n + 1$ distinct points $\mathbf{x}^{(1)}$ to $\mathbf{x}^{(n+1)}$ randomly from the N sample points).

Taking the sample size $N = 10(n + 1)$ gives satisfactory results. In Step 1 of the composite algorithm, if $f^W/f^L < 1 + \varepsilon$ ($\varepsilon > 0$ is a small number), then the global phase may be terminated. Any other stopping criterion may also be used.

18.3.5 Acceptance-Rejection Methods

The *acceptance-rejection* (A-R) methods are *modifications of the multistart algorithm* to improve its efficiency by using ideas from statistical mechanics. In the multistart method, a local minimization is started from each randomly generated point. Thus, the number of local minimizations is very large and many of them converge to the same local minimum point. A strategy to improve this situation is to start the local minimization procedure only when the randomly generated point has a smaller cost function value than that of the local minimum previously obtained. This forces the algorithm to tunnel below irrelevant local minima. This modification, however, has been shown to be inefficient. As a result, the tunneling process has been pursued only by means of deterministic algorithms explained earlier. The acceptance-rejection based methods modify this tunneling procedure. The basic idea is to sometimes start local minimization from a randomly generated point even if it has a higher cost function value than that at a previously obtained local minimum. This is called the *acceptance phase*, which involves calculation of certain probabilities. If the local minimization procedure started from an accepted point produces a local minimum that has higher cost function value than a previously obtained minimum, then the new minimum point is rejected (*rejection phase*). The procedure just described is sometimes called *random tunneling*.

A possible formulation of the *acceptance criterion* to start the local minimization is suggested by the statistical mechanics approach described earlier in Chapter 15, the simulated annealing approach. Thus, the acceptance-rejection methods resemble the simulated annealing approach. The local minimization is started from a point \mathbf{x} only if it has the probability given by

$$p(\mathbf{x}) = \exp\left(\frac{[f(\mathbf{x}) - \bar{f}]_+}{-F}\right) \quad (18.12)$$

where \bar{f} is an estimate of the upper bound of the global minimum, F is a target value for the global minimum, and $[h]_+ = \max(0, h)$. The initial value of F is usually provided by the user, or it may be estimated using a few random points. In this algorithm, unlike simulated annealing, the choice of schedule for reduction of the target level does not prevent convergence. Nevertheless, the schedule is critical for performance of the algorithm. \bar{f} is adjusted at each iteration as the best approximation to the global minimum value. At the start, it may be taken as the smallest cost function value among some randomly generated points, or it can be supplied by the user if a better value is known.

18.3.6 Stochastic Integration

In the *stochastic integration* methods, a suitable stochastic perturbation of the system of equations for the trajectory methods described in Section 18.2.3 is introduced in order to force the trajectory to a global minimum point. This is achieved by monitoring the cost function value along the trajectories. By changing some coefficients in the differential equations, we get different solution processes starting from the same initial point. This idea is similar to simulated annealing but here the differential equation parameter is decreased continuously. We describe the stochastic-integration global minimization method using the steepest descent trajectory. In this, a stochastic perturbation is introduced in Eq. (18.5) in order to increase the chance for the trajectory to reach the global minimum point. The resulting system of stochastic differential equations is given as

$$d\mathbf{x}(t) = -\nabla f(\mathbf{x})dt + \varepsilon(t)d\mathbf{w}(t), \quad \text{with} \quad \mathbf{x}(0) = \mathbf{x}^{(0)} \quad (18.13)$$

where $\mathbf{w}(t)$ is an n -dimensional standard *Wiener process* and $\varepsilon(t)$ is a real function called the *noise coefficient*. In actual implementation, a standard Gaussian distribution is usually used instead of the Wiener process.

Let $\mathbf{x}(t)$ be the solution of Eq. (18.13) starting from $\mathbf{x}^{(0)}$ with a constant noise coefficient $\varepsilon(t) = \varepsilon_0$. Then, as is well known in the field of statistical mechanics, the probability density function of $\mathbf{x}(t)$ approaches the limit density $Z\exp[-2f(x)/\varepsilon_0^2]$, as $t \rightarrow \infty$, where Z is a normalization constant. The limit density is independent of $\mathbf{x}^{(0)}$ and peaks around the global minima of $f(\mathbf{x})$. The peaks become narrower with smaller ε_0 ; i.e., ε_0 is equivalent to the target level F that decreases in the simulated annealing method. In this method, an attempt is made to obtain the global minima by looking at the asymptotic (as $t \rightarrow \infty$) values of a numerically computed sample trajectory of Eq. (18.13), where the noise function $\varepsilon(t)$ is continuous and suitably tends to zero as $t \rightarrow \infty$. In other words, unlike simulated annealing, the target level is lowered continuously. The computational effort in this method can be reduced by observing that a correct numerical computation of the gradient in Eq. (18.13) is not really needed since a stochastic term is added to it. An approximate finite difference gradient may be used instead.

Computing a single trajectory of Eq. (18.13) by decreasing $\varepsilon(t)$ (for $t > 0$) and following the trajectory for a long time to obtain a global solution may not be very efficient. Therefore, in actual implementation an alternative strategy can be used where several trajectories are generated simultaneously. The cost function values along all the trajectories are monitored and compared with each other. A point corresponding to the smallest cost function value on any of the trajectories at any trial is stored. If some trajectories are not progressing satisfactorily, they may be discarded and new ones initiated. As with other stochastic methods, the procedure is executed several times before accepting the best point as the global optimum point.

18.4 Two Local-Global Stochastic Methods

In this section, we describe two stochastic global optimization methods that have both local and global phases. The algorithms have been designed to treat general constraints in the problem explicitly. The algorithms can be viewed as a modification of the multistart procedure but with the ability to learn as the search progresses.

18.4.1 A Conceptual Local-Global Algorithm

As explained in the last section, most stochastic methods have local and global phases. In this subsection, we describe a conceptual algorithm having both of these phases that forms the basis for the two algorithms described in the next two sections.

- Step 1.* Generate a random point $\mathbf{x}^{(0)}$ in the set S_b .
- Step 2.* Check some rejection criteria (discussed later) based on proximity of $\mathbf{x}^{(0)}$ to one of the previous starting points, local minimum points, or rejected points. If a rejection criterion is satisfied, add $\mathbf{x}^{(0)}$ to the set of rejected points, and go to Step 1. Otherwise, execute the local phase by continuing with Step 3.
- Step 3.* Add $\mathbf{x}^{(0)}$ to the set of starting points and find a local minimum \mathbf{x}^* in the feasible set S for the problem.
- Step 4.* Check if \mathbf{x}^* is a new local minimum; if so, add it to the set of local minima, otherwise add $\mathbf{x}^{(0)}$ to the set of rejected points. Go to Step 1.

Steps 1 and 2 constitute the global phase and Steps 3 and 4 constitute the local phase of the algorithm. The basic idea of the algorithm is to explore the entire feasible domain in a systematic way for the global minimum. Bearing in mind that generation of a random point and its evaluation is much cheaper than one local minimization, which may require many function and gradient evaluations, more emphasis is placed on the global phase of the algorithm. The algorithm avoids searching near any local minimum point and all the points leading to it, thus increasing the chance of finding a new local minimum in the unexplored region. To do this, several sets that contain certain types of points are constructed. For example, a set is reserved for all the local minimum points found, and another contains all the starting points for local searches. The following sets, in addition to S and S_b defined earlier, are used in the two algorithms:

- S_* = set of local minima
- S_0 = set of starting points $\mathbf{x}^{(0)}$
- S_r = set of rejected points

A uniformly distributed random point generation scheme over S_b is used so that the entire feasible domain is explored with a uniform probability of finding the global minimum. In Step 1 of the algorithm, the point $\mathbf{x}^{(0)}$ in S_b is accepted because finding a point in S can be a difficult problem (Elwakeil and Arora, 1995). The use of a uniform distribution enables the application of some well-known stopping rules.

The next two sections present the domain elimination and stochastic zooming methods. In both methods, a random point is generated in S_b . Other constraints are ignored at this stage because a local minimization procedure that does not require a feasible starting point can be used. Also, both methods require very little programming effort since the local phase can use existing software. In the algorithms, a modified local phase is used instead of the one given in Step 3: the local search is performed using many subsearches, each one consisting of a few iterations (two or three). Certain criteria, explained in the following sections, are checked after each subsearch to determine if the next subsearch should be started or the local phase terminated.

18.4.2 Domain Elimination Method

The basic idea of this algorithm is to explore the entire feasible domain for the problem in a systematic way for the global minimum. To accomplish this, each local search is attempted from a point that is likely to lead to a new local minimum point. Figure 18-4 shows a conceptual flow diagram for the major steps of the algorithm. The method starts with the selection of a random point from a uniform distribution over the set (Block 2). The point is rejected or accepted based on certain criteria (Block 3); if the point is accepted, then a search for a local minimum is initiated from there (Block 5). If it is rejected, then it is added to the set of rejected points (Block 4) and a new random point is selected. In order to accept or reject a random point, records for the following three types of points are kept: previous starting

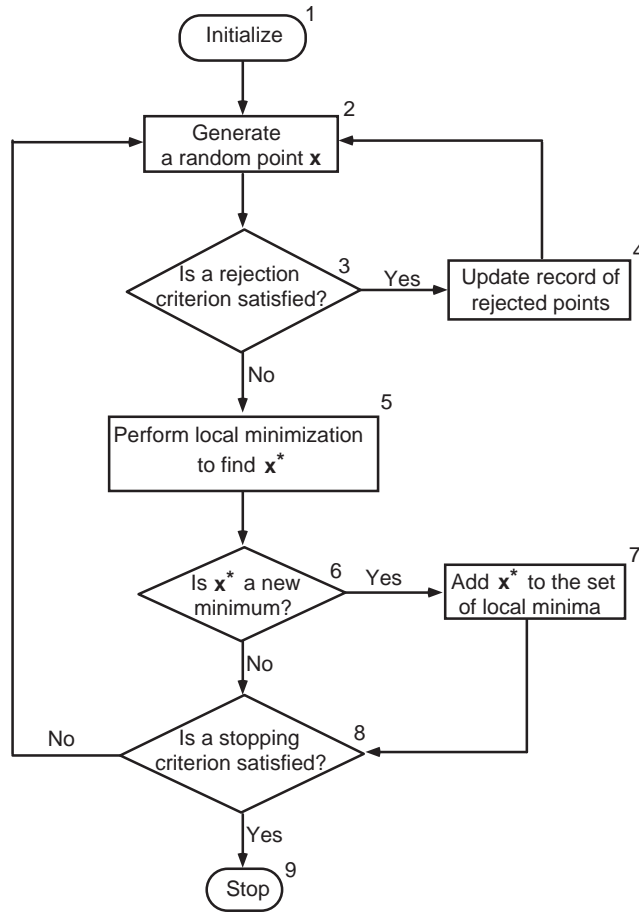


FIGURE 18-4 Flow diagram for domain elimination method.

points for local minimization, local minimum points, and rejected points. A random point is rejected if it is within a critical distance to one of the foregoing sets. The distance between the random point and the points in the sets can be calculated in different ways; the infinity norm being the simplest is suggested. If the point is accepted, a search is initiated for a new local minimum point. The local search process is also monitored, and if the search is going toward a known local minimum point, it is terminated. This is done by checking the closeness of the design points generated during the local search to stored trajectories between previous starting points and the corresponding local minimum points. Thus it is seen that the method *eliminates domains* around the known local minimum points and previously rejected points. At the beginning of the search for a global minimum, most of the random points are accepted as starting points for local minimization. However, near the tail end of the search process, fewer local minimizations are performed.

The following *counters* are used in the algorithm given below:

- c_1 number of elements in S_r , the *set of rejected points*
- c_2 number of rejected points that were near S_r
- c_3 number of elements in S_0 , the *set of starting points*
- c_4 number of elements in S_s , the *set of local minimum points*

- Step 1.* Initialize the sets S_* , S_0 , and S_r . Select a value for the parameter M that specifies the number of local search iterations to be performed to determine an intermediate point \mathbf{x}^M for checking the rejection criteria. Set global iteration counter $i = 0$. Set four counters c_1 , c_2 , c_3 , and c_4 to 0; these are used to keep track of the number of elements in various sets defined earlier.
- Step 2.* If one of the stopping criteria is met, then stop. Otherwise generate a random point \mathbf{x}^R drawn from a uniform distribution over S_b .
- Step 3.* If the current point \mathbf{x}^R is in or near S_* or S_0 (determined using a procedure presented later in Section 18.4.4), then add \mathbf{x}^R to S_r and set $c_1 = c_1 + 1$; else if it is in or near S_r , then set $c_2 = c_2 + 1$; else if \mathbf{x}^R is near a trajectory (a mapping $S_0 \rightarrow S_*$), then add \mathbf{x}^R to S_r and set $c_1 = c_1 + 1$. If any of the four conditions is true, go to Step 2 (this avoids starting the local phase that has a low probability of finding a new minimum). Otherwise add \mathbf{x}^R to S_0 , and set $\mathbf{x}^{(0)} = \mathbf{x}^R$ and $c_3 = c_3 + 1$.
- Step 4.* Execute the local minimization for M iterations to yield an intermediate point \mathbf{x}^M . If \mathbf{x}^M is a local minimum point, then set $\mathbf{x}^* = \mathbf{x}^M$ and go to Step 6. Otherwise, continue.
- Step 5.* If the current point \mathbf{x}^M is in or near S_0 , then add \mathbf{x}^M to S_r and set $c_1 = c_1 + 1$; else if \mathbf{x}^M is in or near S_r , then set $c_2 = c_2 + 1$; else if \mathbf{x}^M is near a trajectory then add \mathbf{x}^M to S_r and set $c_1 = c_1 + 1$. If any of the three conditions is true, then store the trajectory from $\mathbf{x}^{(0)}$ to \mathbf{x}^M and go to Step 2. Otherwise, go to Step 4.
- Step 6.* If \mathbf{x}^* is a new local minimum, then add it to S_* and set $c_4 = c_4 + 1$; otherwise, increment the associated indicator of the number of times \mathbf{x}^* was found.
- Step 7.* Set $i = i + 1$; if i is larger than a specified limit, then stop. Otherwise, go to Step 2.

Stopping Criteria Several stopping criteria can be used and two of them are discussed next. In Step 2 of the algorithm, the procedure is stopped if any of the sets S_0 , S_* , or S_r becomes full based on prespecified limits on the sizes of the sets. The maximum size of a set is determined based on the number of design variables and a confidence level parameter. Since the random points are drawn from a uniform distribution, the number of points generated before stopping the search is proportional to n . The size of set S_* of local minimum points is usually much smaller than the other two sets (since there could be only one minimum for each starting point). In the numerical evaluation presented later, the sizes chosen for current implementation are $10n$ for S_* and $40n$ for S_0 and S_r .

Another criterion is to stop the algorithm if the number of local minima found exceeds the Bayesian estimate for the number of local minima explained earlier in Section 18.3.2, or a specified limit on iterations is exceeded.

18.4.3 Stochastic Zooming Method

This method is an extension of the zooming method described earlier in Section 18.2.2. Recall that as the target level for the cost function gets closer to the global optimum, it becomes difficult to find a feasible point in the set S . Eventually, the modified problem needs to be declared as infeasible to stop the algorithm. To overcome this difficulty, that algorithm is modified by adding a global phase to it to ensure that the set S is reasonably well searched before declaring the modified problem to be infeasible and accepting the previous local minimum as the global minimum. The major difference between this method and the domain elimination method is the addition of the zooming constraint of Eq. (18.4). Therefore that algorithm can be used with some minor modification to keep track of the number of local searches that did not terminate at a feasible minimum point. To do that, the number of iterations to search for a local minimum in Steps 4 and 5 of the algorithm is monitored. If this number exceeds a specified limit, then the local search process is declared to have failed. The

number of such failures is also monitored, and if this number exceeds a specified limit, the algorithm is terminated and the best local minimum is taken as the global minimum point for the problem. In addition to this stopping criterion, if the cost function value reaches a target value F specified by the user [i.e., $f(\mathbf{x}) \leq F$] the algorithm is terminated.

18.4.4 Operations Analysis of Methods

It is seen that the domain elimination and stochastic zooming methods differ from each other primarily in the inclusion of the constraint of Eq. (18.4) in the zooming method. This is an important difference, however, because it changes the behavior of the basic algorithm considerably. This section presents an analysis of the operations and choice of design variable bounds used in both methods. The analysis includes numerical requirements and performance with available implementation alternatives for various rejection criteria used in the algorithm. The following calculations are needed in the algorithms: (1) distance between a point and a set of points, (2) approximation of the trajectory between a starting point and a local minimum, and (3) distance between a point and a trajectory. Each of these calculations can be accomplished with several different procedures. The operations count for each option is necessary for choosing the most efficient procedure.

Checking Proximity of a Point to a Set After generating a random point, one needs to determine if it will yield a new local minimum. For this purpose, the random point \mathbf{x}^R is compared with the points in the three sets S_* , S_0 , and S_r . If it is within a certain critical distance D_{cr} from any point in these sets, it is discarded and a new random point is generated. The same procedure is used for the intermediate point \mathbf{x}^M (Step 4 of the algorithm). Two methods are presented for checking proximity of a point to a set.

Let \mathbf{x}^S be a point belonging to one of the sets mentioned above and \mathbf{x}^M be either a random point or an intermediate point. The first method is to construct a hypersphere of either constant or variable radius around \mathbf{x}^S . The proposed point (\mathbf{x}^R or \mathbf{x}^M) is rejected if it lies inside the hypersphere. This involves calculation of the distance between the two points $D = \|\mathbf{x}^S - \mathbf{x}^M\|$ and the point is rejected if $D \leq D_{cr}$, where D_{cr} is specified as $\alpha\|\mathbf{x}\|$ with $\mathbf{x} = \mathbf{x}^R$ or \mathbf{x}^M and $0.01 \leq \alpha \leq 0.20$. The second method is to construct a hyperprism around \mathbf{x}^S rather than a hypersphere. The proposed point is rejected if it lies inside the hyperprism. In this case, the distance between the two points is not required. Each of the design variables is compared in turn with the corresponding one for the prism's center. If the difference is larger than the corresponding critical value then the rest of the variables need not be compared and the point is accepted. This can be represented by the following pseudocode [let $D_{cr(i)} = \alpha|x_i|$ be a vector with $x_i = (\mathbf{x}^R \text{ or } \mathbf{x}^M)_i$]:

```

for i = 1 to n do
    if  $(\mathbf{x}^S - \mathbf{x}^M)_i \geq D_{cr(i)}$  then accept  $\mathbf{x}^M$ 
end do
reject  $\mathbf{x}^M$ 

```

Based on the operations count, it is seen that the second approach is less expensive.

Trajectory Approximation The random points \mathbf{x}^R selected for starting a local search as well as the intermediate points \mathbf{x}^M during local minimization are examined for proximity to each stored trajectory. A trajectory is the design history from a starting point to the corresponding local minimum point. There could be many trajectories meeting at one local minimum point. The selected point is rejected if it is near any trajectory. This is done to prevent unnecessary minimization steps that would otherwise lead to already known local minima. The trajectory can be approximated using several techniques. The simplest approximation of the trajectory

is a straight line connecting $\mathbf{x}^{(0)}$ and the corresponding \mathbf{x}^* . Experiments have shown that actual trajectories usually do not follow straight lines, especially at the beginning of the search and for nonlinear problems. Other alternatives to approximate the trajectory include: (1) passing a least squares straight line through several points along the trajectory, (2) passing straight line segments through selected points along the trajectory, (3) passing a quadratic curve through three points, (4) passing quadratic segments through groups of three points, and (5) constructing higher-order polynomial or spline approximations. Several issues affect the decision to select the appropriate technique: the number of points needed (which have to be stored), number of operations, and accuracy of the approximation. Any technique other than the straight line approximation requires more intermediate points to be saved and more calculations. Therefore, use of a straight line approximation is suggested.

Distance between a Point and a Trajectory With the linear approximation for the trajectory, the decision whether a point \mathbf{x} lies near the trajectory can be made in several ways. The first procedure is to calculate the internal angle $\mathbf{x}^{(0)} - \mathbf{x} - \mathbf{x}^*$ of the triangle formed by the three points. The point \mathbf{x} is rejected if the angle is larger than a threshold value. The second approach is to calculate an offset distance by generating $\bar{\mathbf{x}}$ as a projection of \mathbf{x} on the line $\mathbf{x}^{(0)} - \mathbf{x}^*$. If $\bar{\mathbf{x}}$ lies outside the line segment $\mathbf{x}^{(0)} - \mathbf{x}^*$, then it is accepted; otherwise the offset length $\mathbf{x} - \bar{\mathbf{x}}$ is calculated. If it is larger than a critical value, then \mathbf{x} is considered far from the trajectory.

Geometrical representation of the triangle method indicates construction of an ellipsoidal body around the line segment $\mathbf{x}^{(0)} - \mathbf{x}^*$. The offset method, on the other hand, can be represented by a cylinder constructed with $\mathbf{x}^{(0)} - \mathbf{x}^*$ as its axis. A point \mathbf{x} is considered close to the trajectory if it lies inside the ellipsoidal body or the cylinder, respectively. The offset method features a uniform critical distance from the linear trajectory, whereas with the triangle method the distance is made smaller near the end points. This means that the critical offset distance in the triangle method is related to the trajectory's length. This makes physical sense because the trajectory's length can be related to the size of the region of attraction for the local minimum. The same effect can be achieved for the cylinder method by requiring the critical offset to be proportional to the length of the line segment $\mathbf{x}^{(0)} - \mathbf{x}^*$ (i.e., $\beta\|\mathbf{x}^{(0)} - \mathbf{x}^*\|$ for some $\beta > 0$). The proportional offset has the advantage of accounting for the problem's scale and, thus, maintains accuracy. Another advantage is that large subdomains get eliminated from larger regions of attractions and vice versa.

A third approach is to construct a truncated cone with the larger base at \mathbf{x}^* and smaller at $\mathbf{x}^{(0)}$, thus allowing a better identification of close regions of attraction. The cone can be constructed by requiring that the critical offset distance be proportional to the distance $\|\bar{\mathbf{x}} - \mathbf{x}^{(0)}\|$. This option requires more calculations than the simple cylinder method.

Based on the operations count, the triangle method is chosen for implementation described later since it uses fewer multiplications. The critical angle in the range $150^\circ \leq \theta \leq 175^\circ$ has shown good performance ($\theta = 170^\circ$ is used in implementation).

Design Variable Constraints In local minimization algorithms, any simple bounds on the design variables can be treated efficiently. Specification of appropriate bounds on the design variables is more important in stochastic global optimization methods compared with other methods. The further apart these bounds are, the larger the number of random points generated in the set S_b . Consequently, the number of local searches performed is increased, which reduces the efficiency. Therefore, the *design variable bounds for a global optimization problem must be chosen carefully* to reflect the nature of the problem. A simple numerical experiment where the allowable range for one of the design variables out of a total of 40 was doubled, showed that the numerical effort to obtain the same global minimum point increased

by 50 percent (Elwakeil and Arora, 1996a). This clearly shows the importance of selecting appropriate bounds on the design variables in global optimization.

18.5 Numerical Performance of Methods

Various concepts and aspects of global optimization methods have been described in this chapter. Details of some of the algorithms have been presented to give a flavor of the computations that are needed to solve the problem. It is seen that solving a global optimization problem is a challenge from a computational viewpoint, especially when a true global minimum is required. The main reason is that *even if the global minimum point has been reached during the search process, it is not possible to recognize this fact*, that is, there is no definite stopping criterion. Therefore the search process needs to be continued and the algorithm needs to be executed repeatedly to ensure that the global minimum point has not been missed. In other words, the entire feasible set needs to be thoroughly searched, implicitly or explicitly. For practical applications, however, improved local minima or improved feasible designs are acceptable. In that case, reasonably efficient and effective computational algorithms are available or can be devised to achieve this objective. In addition, *many algorithms can be implemented on parallel processors to reduce the “wall clock” time* to solve practical applications.

In this section, we summarize features of the methods described earlier. Numerical performance of some of the methods is described using a limited set of test problems to gain insights into the type of computation needed and the behavior of the methods (Elwakeil and Arora, 1996a,b). Several structural design problems have also been devised and solved to study that class of problems for global optimization.

18.5.1 Summary of Features of Methods

It is *difficult to recommend a single global optimization method for all applications*. Selection of a method depends on characteristics of the problem and what is desired. For example, if all the local minima are desired, then the tunneling or zooming method is not suitable. If the problem has discrete variables and the functions are not differentiable, then a method that requires and uses gradients is not suitable. If an absolute guarantee of a global solution is desired, then certain methods that do not guarantee this are not appropriate. Therefore, it is suggested that the problem characteristics and requirements be analyzed before selecting an algorithm for global optimization. Table 18-1 summarizes the characteristics of various algorithms described in this chapter. Also be aware that no matter which algorithm is selected, the computational effort to reach a solution point is substantial. Therefore, one must be willing and able to bear the enormous cost of finding an estimate of a global solution for the problem. The table summarizes the following characteristics of various global optimization algorithms:

1. Classification of method: deterministic (D) or stochastic (S).
2. Ability of the method to solve discrete problems; it is desirable that the method be able to solve discrete problems.
3. Ability to treat general constraints explicitly; this is a desirable feature.
4. Ability to find all the local minima; this depends on the desire of the user.
5. Use of local-global phases; methods using both phases are generally more reliable and efficient.
6. Need for gradients; if a method definitely needs gradients of the functions, then its applicability is limited to continuous problems only.

TABLE 18-1 Characteristics of Global Optimization Methods

<i>Method</i>	<i>Can solve discrete problems?</i>	<i>General constraints?</i>	<i>Tries to find all \mathbf{x}^*?</i>	<i>Phases</i>	<i>Needs gradients?</i>
Covering (D)	No	No	Yes	G	1
Zooming (D)	Yes ¹	Yes	No	L	1
Generalized descent (D)	No	No	No	G	Yes
Tunneling (D)	No	Yes	No	L + G	1
Multistart (S)	Yes ¹	Yes	Yes	L + G	1
Clustering (S)	Yes ¹	Yes	Yes	L + G	1
Controlled random (S)	Yes	No	No	L + G	No
Simulated annealing (S)	Yes	No	No	G	No
Acceptance-rejection (S)	Yes ¹	Yes	No	G	No
Stochastic integration (S)	No	No	No	G	No
Genetic (S)	Yes	No	No	G	No
Stochastic zooming (S)	Yes ¹	Yes	No	L + G	1
Domain elimination (S)	Yes ¹	Yes	Yes	L + G	1

D: deterministic methods; S: stochastic methods; G: global phase; L: local phase.

¹Depends on the local minimization procedure used.

18.5.2 Performance of Some Methods Using Unconstrained Problems

As a first numerical performance study, the following four methods were implemented (Elwakeil and Arora, 1996a): covering method, acceptance-rejection method (A-R), controlled random search (CRS), and simulated annealing (SA). The numerical tests were performed on 29 unconstrained problems available in the literature. The problems had one to six design variables and only explicit bounds on them. Global solutions for the problems were known.

Based on the results, it was concluded that the covering methods were not practical because of their inefficiency for problems with $n > 2$. The methods required very large computational effort. Also, it was difficult to generate a good estimate for the Lipschitz constant that is needed in the algorithm. Both A-R and CRS methods performed better than simulated annealing and the covering method. The fact that the A-R method does not include any stopping criterion makes it undesirable for practical applications. The method worked efficiently on test problems because it was stopped upon finding the known global optimum point. The CRS method contains a stopping criterion and is more efficient compared with other methods. An attempt to treat general constraints explicitly in the CRS method was not successful because constraint violations could not be corrected in reasonable computational effort.

18.5.3 Performance of Stochastic Zooming and Domain Elimination Methods

In another study, the stochastic zooming method (ZOOM) and the domain elimination (DE) method were also implemented (in addition to CRS and SA), and their performance was evaluated using 10 mathematical programming test problems (Elwakeil and Arora, 1996a). The test problems included constrained as well as unconstrained problems. Even though most

engineering application problems are constrained, it is beneficial to test performance of the algorithms on the unconstrained problems as well. The CRS method could be used only for unconstrained problems. It is noted, however, that the problems classified as unconstrained still include simple bounds on the design variables. The sequential quadratic programming (SQP) method was used in all local searches performed in the ZOOM and DE methods. For ZOOM, the percent reduction required from one local minimum to the next was set arbitrarily to 15 percent [i.e., $\gamma = 0.85$ in Eq. (18.4)] for all the test problems.

The 10 test problems used in the study had the following characteristics: 4 problems had no constraints, the number of design variables varied from 2 to 15, the total number of general constraints varied from 2 to 29, 2 problems had equality constraints, all problems had 2 or more local minima, 2 problems had 2 global minima and 1 had 4, 1 had global minimum as 0, and 4 had negative global minimum values.

To compare performance of different algorithms, each of the test problems was solved five times and averages for the following evaluation criteria were recorded: number of random starting points, number of local searches performed, number of iterations used during the local search, number of local minima found by the method, cost function value of the best local minimum (the global minimum), total number of calls for function evaluations, and CPU time used.

Because a random point generator with a random seed was used, the performance of the algorithms changed each time they were executed. The seed is automatically chosen based on the wall clock time. The results differed in the number of local minima found as well as for the other evaluation criteria.

DE found the global solution for 9 out of 10 problems, whereas ZOOM found a global minimum for 7 out of 10 problems. In general, DE found more local minima than ZOOM did. This is attributed to the latter requiring a reduction in the cost function value after each local minimum was found. As noted earlier, ZOOM is designed to “tunnel” under some minima with relatively close cost function values.

In terms of the number of function evaluations and CPU time, DE was cheaper than ZOOM. This was because the latter performed more local iterations for a particular search without finding a feasible solution. On the other hand, the number of iterations during a local search performed in DE was smaller since it could find a solution in most cases.

The CPU time needed by CRS was considerably smaller than that for other methods even with a larger number of function evaluations. This was due to the use of a local search procedure that did not require gradients or line search. However, the method is applicable to only unconstrained problems.

Simulated annealing (SA) failed to locate the global minimum for six problems. For the successful problems, the CPU time required was three to four times that for DE. The tests also showed that there was a drastic increase in the computational effort for the problems as the number of design variables increased. Therefore, that implementation of SA was considered inefficient and unreliable compared with that of both DE and ZOOM. It is noted that the SA may be more suitable for problems with discrete variables only.

18.5.4 Global Optimization of Structural Design Problems

The DE and ZOOM methods have also been applied to structural design problems to find global solutions for them (Elwakeil and Arora, 1996b). In this section, we summarize and discuss results of that study which used the following 6 structures: a 10-bar cantilever truss, a 200-bar truss, a 1-bay–2-story frame, a 2-bay–6-story-frame, a 10-member cantilever frame, and a 200-member frame. These structures have been used previously in the literature to test various algorithms for local minimization (Haug and Arora, 1979). A variety of constraints were imposed on the structures: constraints and other requirements given in the Specification of the American Institute of Steel Construction (AISC, 1989), Aluminum Association

Specifications (AA, 1986), displacement constraints, and constraints on the natural frequency of the structure. Some of the structures were subjected to multiple loading cases. For all problems, the weight of the structure was minimized. Using these 6 structures, 28 test problems were devised by varying the cross-sectional shape of members to hollow circular tubes or I sections, and changing the material from steel to aluminum. The number of design variables varied from 4 to 116, the number of stress constraints varied from 10 to 600, the number of deflection constraints varied from 8 to 675, and the number of local buckling constraints for the members varied from 0 to 72. The total number of general inequality constraints varied from 19 to 1276. These test problems can be considered to be large compared with the ones used in the previous section.

Detailed results using DE and ZOOM can be found in Elwakeil (1995). Each problem was solved five times with a different seed for the random number generator. The five runs were then combined and all the optimum solutions found were stored.

It was observed that all the six structures tested possessed many local minima. ZOOM found only one local minimum for each problem (except two problems). For most of the problems, the global minimum was found with the first random starting point. Therefore, other local minima were not found since they had a higher cost function value. DE found many local minima for most of the problems except for one problem that turned out to be infeasible. The method did not find all the local minima in one run because of the imposed limit on the number of random starting designs. From the recorded CPU times, it was difficult to draw a general conclusion about the relative efficiency of the two methods because for some problems one method was more efficient and for the remaining the second method was more efficient. However, each of the methods can be useful depending on the requirements. If only the global minimum is sought, then ZOOM can be used. If all or most of the local minima are wanted, then DE should be used. The zooming method can be used to determine lower-cost practical designs by appropriately selecting the parameter γ in Eq. (18.4).

Some problems showed only a small difference between weights for the best and the worst local minima. This indicates a flat feasible domain perhaps with small variations in the weight which results in multiple global minima. One of the problems was infeasible because of an unreasonable requirement for the natural frequency to be no less than 22 Hz. However, when the constraint was gradually relaxed, a solution was found at a value of 17 Hz.

It is clear that the designer's experience and knowledge about the problems, and the design requirements can affect performance of the global optimization algorithms. For example, by setting a correct limit on the number of local minima desired, the computational effort of the domain elimination method can be reduced substantially. For the zooming method, the computational effort will be reduced if the parameter γ in Eq. (18.4) is selected judiciously. In this regards, it may be possible to develop a strategy to automatically adjust the value of γ dynamically during local searches. This will avoid the infeasible problems which constitute a major computational effort in the zooming method. Also, a realistic value for F , the target value for the global minimum cost function, would improve efficiency of the method.

Exercises for Chapter 18*

Calculate a global minimum point for the following problems.

18.1 (Branin and Hoo, 1972)

minimize

$$f(\mathbf{x}) = \left(4 - 2.1x_1^2 + \frac{1}{3}x_1^4 \right) x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$$

subject to

$$-3 \leq x_1 \leq 3$$

$$-2 \leq x_2 \leq 2$$

18.2 (Lucidi and Piccion, 1989)

minimize

$$f(\mathbf{x}) = \frac{\pi}{n} \left\{ 10 \sin^2(\pi x_1) + \sum_{i=1}^{n-1} [(x_i - 1)^2 (1 + 10 \sin^2(\pi x_{i+1}))] + (x_n - 1)^2 \right\}$$

subject to

$$-10 \leq x_i \leq 10; \quad i = 1 \text{ to } 5$$

18.3 (Walster *et al.*, 1984)

minimize

$$f(\mathbf{x}) = \sum_{i=1}^{11} \left[a_i - x_1 \frac{b_i^2 - b_i x_2}{b_i^2 + b_i x_3 + x_4} \right]$$

subject to

$$-2 \leq x_i \leq 2; \quad i = 1 \text{ to } 4$$

where the coefficients (a_i, b_i) ($i = 1$ to 11) are given as follows:

(0.1975, 4), (0.1947, 2), (0.1735, 1), (0.16, 0.5), (0.0844, 0.25), (0.0627, 0.1667),
(0.0456, 0.125), (0.0342, 0.1), (0.0323, 0.0833), (0.0235, 0.0714), (0.0246, 0.0625).

18.4 (Evtushenko, 1974)

minimize

$$f(\mathbf{x}) = - \left[\sum_{i=1}^6 \frac{1}{6} \sin 2\pi \left(x_i + \frac{i}{5} \right) \right]^2$$

subject to

$$0 \leq x_i \leq 1; \quad i = 1 \text{ to } 6$$

18.5 minimize

$$f(\mathbf{x}) = 2x_1 + 3x_2 - x_1^3 - 2x_2^2$$

subject to

$$\frac{1}{6}x_1 + \frac{1}{2}x_2 - 1.0 \leq 0$$

$$\frac{1}{2}x_1 + \frac{1}{5}x_2 - 1.0 \leq 0$$

$$x_1, x_2 \geq 0$$

18.6 (Hock and Schittkowski, 1981)

minimize

$$f(\mathbf{x}) = \sum_{i=1}^{99} f_i^2(\mathbf{x})$$

$$f_i(\mathbf{x}) = -\frac{i}{100} + \exp\left(-\frac{1}{x_1}(u_i - x_2)^{x_3}\right)$$

$$u_i = 25 + [-50 \ln(0.01i)]^{2/3}; \quad i = 1 \text{ to } 99$$

subject to

$$0.1 \leq x_1 \leq 100, \quad 0.0 \leq x_2 \leq 25.6, \quad 0.0 \leq x_3 \leq 5$$

18.7 (Hock and Schittkowski, 1981)

minimize

$$f(\mathbf{x}) = (x_1^2 - x_2^2) + (x_2^2 - x_3^2) + (x_3^2 - x_4^2) + (x_4^2 - x_5^2)$$

subject to

$$x_1 + x_2^2 + x_3^3 - 3 = 0$$

$$x_1 - x_3^2 + x_4 - 1 = 0$$

$$x_1 x_5 - 1 = 0$$

18.8 (Hock and Schittkowski, 1981)

minimize

$$\begin{aligned} f(\mathbf{x}) = & -75.196 + b_1 x_1 + b_2 x_1^3 - b_3 x_1^4 + b_4 x_2 - b_5 x_1 x_2 + b_6 x_2 x_1^2 \\ & + b_7 x_1^4 x_2 - b_8 x_2^2 + c_1 x_2^3 - c_2 x_2^4 + 28.106/(x_2 + 1) + c_3 x_1^2 x_2^2 \\ & + c_4 x_1^3 x_2^2 - c_5 x_1^3 x_2^3 - c_6 x_1 x_2^2 + c_7 x_1 x_2^3 + 2.8673 \exp\left(\frac{x_1 x_2}{2000}\right) - c_8 x_1^3 x_2 \end{aligned}$$

subject to

$$x_1 x_2 - 700 \geq 0$$

$$x_2 - x_1^2/125 \geq 0$$

$$(x_2 - 50)^2 - 5(x_1 - 55) \geq 0$$

$$0 \leq x_1 \leq 75, \quad 0 \leq x_2 \leq 65$$

where the parameters (b_i, c_i) ($i = 1$ to 8) are given as

(3.8112E+00, 3.4604E-03), (2.0567E-03, 1.3514E-05),
 (1.0345E-05, 5.2375E-06), (6.8306E+00, 6.3000E-08),
 (3.0234E-02, 7.0000E-10), (1.2814E-03, 3.4050E-04),
 (2.2660E-07, 1.6638E-06), (2.5645E-01, 3.5256E-05).

18.9 (Hock and Schittkowski, 1981)

minimize

$$f(\mathbf{x}) = x_1 x_4 (x_1 + x_2 + x_3) + x_3$$

subject to

$$x_1 x_2 x_3 x_4 - 25 \geq 0$$

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 - 40 = 0$$

$$1 \leq x_i \leq 5; \quad i = 1 \text{ to } 4$$

18.10 (Hock and Schittkowski, 1981)

minimize

$$f(\mathbf{x}) = \sum_{k=0}^4 (2.3x_{3k+1} - (1.0E-4)x_{3k+1}^3 + 1.7x_{3k+2} + (1.0E-4)x_{3k+2}^2 + 2.2x_{3k+3} + (1.5E-4)x_{3k+3}^2)$$

subject to

$$0 \leq x_{3j+1} - x_{3j-2} + 7 \leq 13; \quad j = 1 \text{ to } 4$$

$$0 \leq x_{3j+3} - x_{3j-2} + 7 \leq 14; \quad j = 1 \text{ to } 4$$

$$0 \leq x_{3j+3} - x_{3j} + 7 \leq 13; \quad j = 1 \text{ to } 4$$

$$x_1 + x_2 + x_3 - 60 \geq 0$$

$$x_4 + x_5 + x_6 - 50 \geq 0$$

$$x_7 + x_8 + x_9 - 70 \geq 0$$

$$x_{10} + x_{11} + x_{12} - 85 \geq 0$$

$$x_{13} + x_{14} + x_{15} - 105 \geq 0$$

and the bounds are ($k = 1$ to 4):

$$\begin{aligned} 8.0 \leq x_1 \leq 21.0, & \quad 43.0 \leq x_2 \leq 57.0, & \quad 3.0 \leq x_3 \leq 16.0, & \quad 0.0 \leq x_{3k+1} \leq 90.0, \\ 0.0 \leq x_{3k+2} \leq 120.0, & & \quad 0.0 \leq x_{3k+3} \leq 60.0. & \end{aligned}$$

Find all the local minimum points for the following problems and determine a global minimum point.

- | | | | | | |
|-------|----------------|-------|---------------|-------|---------------|
| 18.11 | Exercise 18.1 | 18.12 | Exercise 18.2 | 18.13 | Exercise 18.3 |
| 18.14 | Exercise 18.4 | 18.15 | Exercise 18.5 | 18.16 | Exercise 18.6 |
| 18.17 | Exercise 18.7 | 18.18 | Exercise 18.8 | 18.19 | Exercise 18.9 |
| 18.20 | Exercise 18.10 | | | | |

Appendix A Economic Analysis

The main body of this textbook describes promising analytical and numerical techniques for engineering design optimization. *This appendix departs from the main theme and contains an introduction to engineering decision making based on economic considerations.* More detailed treatment of the subject can be found in texts by Grant and coworkers (1982) and Blank and Tarquin (1983).

A.1 Time Value of Money

Engineering systems are designed to perform specific tasks. Usually many alternative designs can perform the same task. The question is, which one of the alternatives is the best? Several factors such as precedents, social environment, aesthetic, economic, and psychological values can influence the final selection. This appendix considers only the economic factors influencing the selection of an alternative.

Economic problems are an integral part of engineering because engineers are sensitive to the direct cost of a design. They must anticipate maintenance and operating costs. Future economic conditions must also be taken into account in the decision-making process. *We shall discuss ways to measure the value of money to enable comparisons of alternative designs.* The following notation is used:

n = number of interest periods, e.g., months, years.

i = return per dollar per period; note that i is not the annual interest rate. This shall be further explained in examples.

P = value (or sum) of money at the present time, in dollars.

S_n = final sum after n periods or n payments from the present date, in dollars.

R = a series of consecutive, equal, end-of-period amounts of money—payment or receipt; e.g., dollars per month, dollars per year, and so on.

It is important to understand the notation and the meaning of the symbols to correctly interpret and solve the examples and the exercises. For example, i must be interpreted as the rate of return per dollar per period and not the annual interest rate, and R is the end-of-period

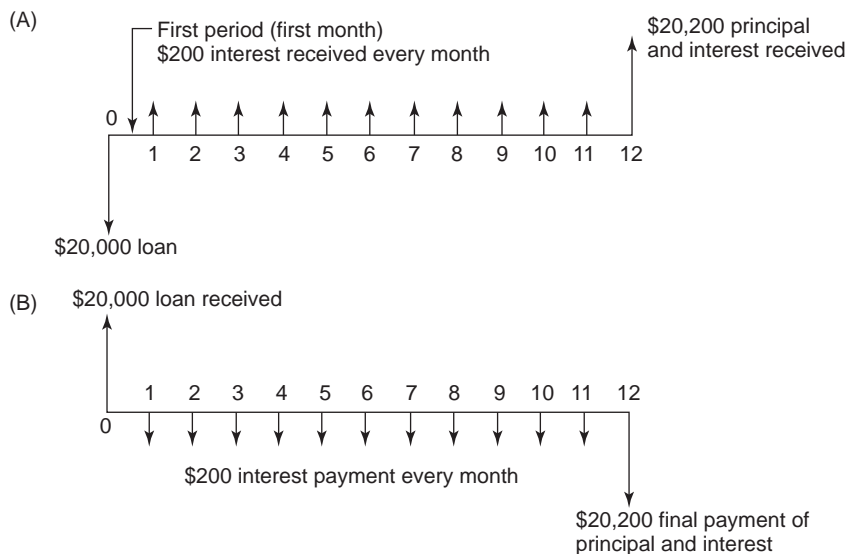


FIGURE A-1 Cash flow diagrams. (A) Lender's cash flow. (B) Borrower's cash flow.

amount and not at the beginning of the period. *It is important to note that we shall quote the annual interest rate in examples and exercises; using that one can calculate i .*

A.1.1 Cash Flow Diagrams

A cash flow diagram is a pictorial representation of cash receipts and disbursements. These diagrams are helpful in solving problems of economic analysis. Once a correct cash flow diagram for the problem has been drawn, it is a simple matter of using proper interest formulas to perform calculations. In this section, we introduce the idea of a cash flow diagram.

Figure A-1 gives a cash flow diagram from two points of view—the lender's and the borrower's. In the diagram, a person has borrowed a sum of \$20,000 and promises to pay it back in 1 year, with simple interest paid every month. The annual interest rate is 12 percent. Therefore, \$200 is paid as interest at the end of every month and \$20,000 principal is paid at the end of the 12th month. *Note that vertical lines with arrows pointing downward imply disbursements and with arrows pointing upward imply receipts.* Also, disbursements are shown below and receipts above the horizontal line.

A.1.2 Basic Economic Formulas

Consider an investment of P dollars that returns i dollars per dollar per period. The return at the end of the first period is iP , and the original investment increases to $(1 + i)P$. This sum is reinvested and returns $i(1 + i)P$ at the end of the next period so that the original amount is worth $(1 + i)^2P$, and so on. If the process is continued for n periods, an original investment P will increase to the final sum S_n , given by

$$S_n = (1 + i)^n P = [\text{spcaf}(i, n)]P \quad (\text{A.1})$$

where $\text{spcaf}(i, n)$ is called the *single payment compound amount factor*.

EXAMPLE A.1 Use of Single Payment Compound Amount Factor

Consider an investment of \$1000 at an annual interest rate of 9 percent compounded monthly. Calculate the final sum at the end of 2 and 4 years.

Solution. For the given annual interest rate, the rate of return per dollar per month is $i = 0.09/12 = 0.0075$. The final sum on an investment of \$1000 at the end of 2 years ($n = 24$) using the single payment compound amount factor of Eq. (A.1) will be

$$\begin{aligned} S_{24} &= \text{spcf}(0.0075, 24)(1000) \\ &= (1 + 0.0075)^{24}(1000) = (1.19641)(1000) = \$1196.41 \end{aligned}$$

and at the end of 4 years ($n = 48$) it will become

$$\begin{aligned} S_{48} &= (1 + 0.0075)^{48}(1000) \\ &= (1.43141)(1000) = \$1431.41 \end{aligned}$$

A future payment S_n made at the end of the n th period has an equivalent present worth P , which can be calculated by inverting Eq. (A.1) as

$$P = (1 + i)^{-n} S_n = [\text{sppwf}(i, n)]S_n \quad (\text{A.2})$$

where $\text{sppwf}(i, n)$ is called the *single payment present worth factor*. Note from Eq. (A.1) that $\text{sppwf}(i, n)$ is the reciprocal of $\text{spcf}(i, n)$.

EXAMPLE A.2 Use of Single Payment Present Worth Factor

Consider the case of a person who wants to borrow some money from the bank but can pay back only \$10,000 at the end of 2 years. How much can the bank lend if the prevailing annual interest rate is 12 percent compounded monthly?

Solution. Using the given rate of interest, the rate of return per dollar per period for this example is $i = 0.12/12 = 0.01$. Using the single payment present worth factor of Eq. (A.2), the present worth of \$10,000 paid at the end of 2 years ($n = 24$) is given as $P = [\text{sppwf}(0.01, 24)](10,000)$:

$$\begin{aligned} P &= (1 + 0.01)^{-24}(10,000) \\ &= 0.787566(10,000) = \$7876.66 \end{aligned}$$

Thus, the bank can lend only \$7876.66 at the present time.

Consider a sequence of n uniform periodic payments R made at the end of each period. The first payment made at the end of the first period earns interest over $(n - 1)$ periods. Therefore, from Eq. (A.1), it is equivalent to an amount $(1 + i)^{n-1}R$ at the end of the n th period. The second payment made at the end of the second period earns interest over $(n - 2)$ periods and is worth $(1 + i)^{n-2}R$ at the end of the n th period; and so on. This sequence of payments is equivalent to a sum S_n , given by the finite geometric series,

$$\begin{aligned} S_n &= (1+i)^{n-1}R + \dots + (1+i)R + R \\ &= [(1+i)^{n-1} + \dots + (1+i) + 1]R \\ &= (1/i)[(1+i)^n - 1]R = [\text{uscaf}(i, n)]R \end{aligned} \quad (\text{A.3})$$

where $\text{uscaf}(i, n)$ is the *uniform series compound amount factor*. Likewise, a future sum S_n can be expressed as an equivalent series of uniform payments R by inverting the expression in Eq. (A.3):

$$R = \frac{iS_n}{[(1+i)^n - 1]} = \text{sfd}(i, n)S_n \quad (\text{A.4})$$

where $\text{sfd}(i, n)$ is called the *sinking fund deposit factor*.

It is important to note in Eqs. (A.3) and (A.4) that

1. n is the number of interest periods and the first payment occurs at the end of the first period.
2. The final sum S_n at the end of the n th period includes the final n th payment.

EXAMPLE A.3 Use of Uniform Series Compound Amount Factor

Consider the case of a person who decides to deposit \$50 at the end of each month for the next 10 years. The prevailing annual rate of interest is 9 percent compounded monthly. How much will have accumulated at the end of a 10-year period?

Solution. Since the interest is compounded monthly, the value of i for this problem is $0.09/12 = 0.0075$. Other data are: $R = \$50$ and $n = 120$. We need to determine S_{120} . Using the uniform series compound amount factor of Eq. (A.3), the final sum S_{120} is given as $S_{120} = [\text{uscaf}(0.0075, 120)](50)$:

$$\begin{aligned} S_{120} &= \frac{[(1 - 0.0075)^{120} - 1]}{0.0075}(50) \\ &= (193.51428)(50) = \$9675.71 \end{aligned}$$

Note that the final sum S_{120} includes the final payment made at the end of the tenth year.

EXAMPLE A.4 Use of Sinking Fund Deposit Factor

A person promises to pay the bank \$10,000 at the end of 2 years. How much money can the bank lend per month if the annual interest rate is 12 percent compounded monthly?

Solution. Since the annual interest rate is 12 percent, the rate of return per dollar per period for this problem is $i = 0.12/12 = 0.01$. Using the sinking fund deposit factor of Eq. (A.4), the amount received at the end of each month is given as $R = [\text{sfd}f(0.01, 24)](10,000)$:

$$\begin{aligned} R &= \frac{0.01}{[(1+0.01)^{24} - 1]}(10,000) \\ &= (0.037073)(10,000) = \$370.73 \end{aligned}$$

Note that the first payment occurs at the end of the first month. Also, the final payment from the bank occurs at the end of 2 years and at that time a payment of \$10,000 must also be made to the bank.

The sequence of n payments R made at the end of each period can also be expressed as a present worth P . Combining Eqs. (A.2) and (A.3) yields

$$P = (1/i)[1 - (1+i)^{-n}]R = [\text{uspwf}(i, n)]R \quad (\text{A.5})$$

where $\text{uspwf}(i, n)$ is called the *uniform series present worth factor*.

Finally, expressing a present amount P as an equivalent sequence of n uniform payments made at the end of each period gives [from Eq. (A.5)]:

$$R = \frac{iP}{[1 - (1+i)^{-n}]} = [\text{crf}(i, n)]P \quad (\text{A.6})$$

where $\text{crf}(i, n)$ is called the *capital recovery factor*. Table A-1 summarizes all the factors.

TABLE A-1 Interest Formulas

To find	Given	Multiply by
S_n	P	Single payment compound amount factor ^a (spcaf), $(1+i)^n$
P	S_n	Single payment present worth factor (sppwf), $(1+i)^{-n}$
S_n	R	Uniform series compound amount factor (uscaf), $\frac{1}{i}[(1+i)^n - 1]$
R	S_n	Sinking fund deposit factor (sfd), $\frac{i}{[(1+i)^n - 1]}$
P	R	Uniform series present worth factor (uspwf), $\frac{1}{i}[1 - (1+i)^{-n}]$
R	P	Capital recovery factor (crf), $\frac{i}{[1 - (1+i)^{-n}]}$

^aThat is, $S_n = [\text{spcaf}(i, n)]P = (1+i)^n P$.

EXAMPLE A.5 Use of Uniform Series Present Worth Factor

A person promises to pay the bank \$100 per month for the next 2 years. If the annual interest rate is 15 percent compounded monthly, how much can the bank afford to lend at the present?

Solution. Since the interest is compounded monthly, the value of i for this example is $0.15/12 = 0.0125$. Using the uniform series present worth factor of Eq. (A.5), the present worth of \$100 paid at the end of each month beginning with the first one and ending with the 24th, is $P = [\text{uspwf}(0.0125, 24)](100)$:

$$\begin{aligned} P &= [1 - (1 + 0.0125)^{-24}](100)/0.0125 \\ &= (20.6242)(100) = \$2062.42 \end{aligned}$$

EXAMPLE A.6 Use of Capital Recovery Factor

A person puts \$10,000 in the bank and would like to withdraw a fixed sum of money at the end of each month over the next 2 years until the fund is depleted. How much can be withdrawn every month at an annual interest rate of 9 percent compounded monthly?

Solution. Since the interest is compounded monthly, $i = 0.09/12 = 0.0075$. Using the capital recovery factor of Eq. (A.6), we can find the amount that can be withdrawn at the end of every month for the next 2 years as $R = [\text{crf}(0.0075, 24)](10,000)$:

$$\begin{aligned} R &= \frac{0.0075(10,000)}{[1 - (1 + 0.0075)^{-24}]} \\ &= (0.045685)(10,000) = \$456.85 \end{aligned}$$

A.2 Economic Bases for Comparison

The formulas given in Table A.1 can be employed in making economic comparisons of alternatives. *Two methods of comparison commonly used are the annual cost (AC) and the present worth (PW) methods.* We shall describe both methods. It is important to realize that the same conditions must be used to compare various alternatives. However, the annual base method allows us to compare easily alternatives having different life spans. This will be illustrated with examples. Also, both methods lead to the same conclusion so either one may be used to compare alternatives.

Sign Convention A note about the *sign convention* used in comparing alternatives is in order. When most of the transactions involve disbursements or costs, a positive sign is used for costs and a negative sign for receipts in calculating annual costs or present worths. In that case, salvage is considered as a negative cost and any other income is also given a negative sign. With this sign convention, present worth greater than zero actually implies present cost, so an alternative with smaller present worth is to be preferred. If present worth has a negative sign, then it actually represents income. Also, in this case, an alternative with smallest present worth taking into account the algebraic sign is to be preferred, i.e., an alternative with the largest numerical value for the present worth. The final selection of an alternative does

not depend on the sign convention used in calculations so any consistent convention can be used. However, we shall use the preceding sign convention in all calculations.

A.2.1 Annual Base Comparisons

An annual base comparison reduces all revenues and expenditures over the selected time to an equivalent annual value. Recall that a positive sign will be used for costs and a negative sign for income. Therefore, the alternative with lower cost is to be preferred.

EXAMPLE A.7 Alternate Designs

A design project has two options, A and B. Option A will cost \$280,000 and Option B \$250,000. Annual operating and maintenance paid at the end of each year will be \$8000 for A and \$10,000 for B. Using the *annual cost (AC) method* of comparison with a 12 percent interest rate, which option should be chosen if both have a 50-year life with no salvage?

Solution. The cash flow diagrams for the two options are shown in Fig. A-2. The annual cost of Option A is the sum of the annual maintenance cost and the equivalent uniform payment of the initial cost (\$280,000). The initial cost can be converted to equivalent yearly payment using the capital recovery factor. Thus, the annual cost (AC_A) of Option A is given as

$$\begin{aligned} AC_A &= 280,000 \text{ crf } (0.12, 50) + 8000 \\ &= \$41,716.67 \end{aligned}$$

since $\text{crf } (0.12, 50) = 0.12042$. Similarly, for Option B, the annual cost is

$$\begin{aligned} AC_B &= 250,000 \text{ crf } (0.12, 50) + 10,000 \\ &= \$40,104.17 \end{aligned}$$

On this basis Option B is cheaper.

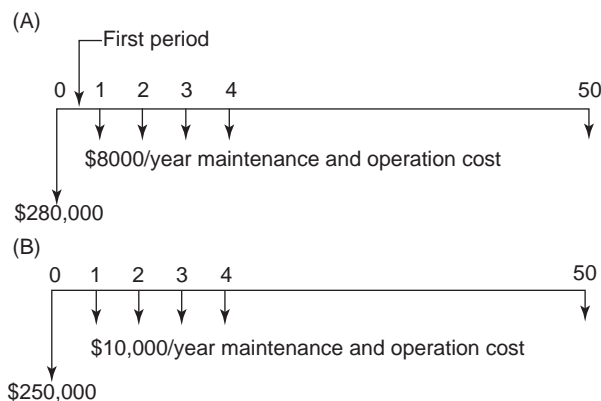


FIGURE A-2 Cash flow diagrams for alternate designs of Example A.7. (A) Option A. (B) Option B.

EXAMPLE A.8 Alternate Power Stations

Three companies have submitted bids shown in the following table for the design and operation of a temporary power station that will be used for 4 years. Which design should be used by annual base comparison if a 15 percent return is required and if all the equipment can be resold after 4 years for 30 percent of the initial cost?

	A	B	C
Initial cost (\$)	5000	6000	7500
Annual cost (\$)	2500	2000	1800

Solution. This example is slightly different from Example A.7 in that the salvage value must also be included while calculating the annual cost. Salvage is an income received at the end of the project. This future sum must be converted to an equivalent yearly income and subtracted from the expenses. A future sum is converted to an equivalent yearly value using the sinking fund deposit factor (sdfd). Therefore, the annual cost of Bid A is given as

$$\begin{aligned}AC_A &= 5000 \text{ crf } (0.15, 4) + 2500 - 0.3(5000) \text{ sdfd } (0.15, 4) \\ &= \$3950.93\end{aligned}$$

since $\text{crf } (0.15, 4) = 0.35027$ and $\text{sdfd } (0.15, 4) = 0.20027$. Similarly, the annual costs of Bids B and C are

$$\begin{aligned}AC_B &= 6000 \text{ crf } (0.15, 4) + 2000 - 0.3(6000) \text{ sdfd } (0.15, 4) \\ &= \$3741.11\end{aligned}$$

$$\begin{aligned}AC_C &= 7500 \text{ crf } (0.15, 4) + 1800 - 0.3(7500) \text{ sdfd } (0.15, 4) \\ &= \$3976.39\end{aligned}$$

Based on these calculations, Bid B is the cheapest option.

EXAMPLE A.9 Alternate Quarries

A company can purchase either of the two mineral quarries. Quarry A costs \$600,000, is estimated to last 12 years, and would have a land-salvage value at the end of 12 years of \$120,000. Digging and shipping operations would cost \$50,000 per year. Quarry B costs \$900,000, would last 20 years, and would have a salvage value of \$60,000. Digging and shipping would cost \$40,000 per year. Which quarry should be purchased? Use the annual base method with an interest rate of 15 percent. Assume that similar quarries will be available in the future.

Solution. Note for the example that the life spans of Quarries A and B are different. This causes no problem when the annual base method is used. However, with the present worth method of the next section, we shall have to somehow use the same life spans for the two options.

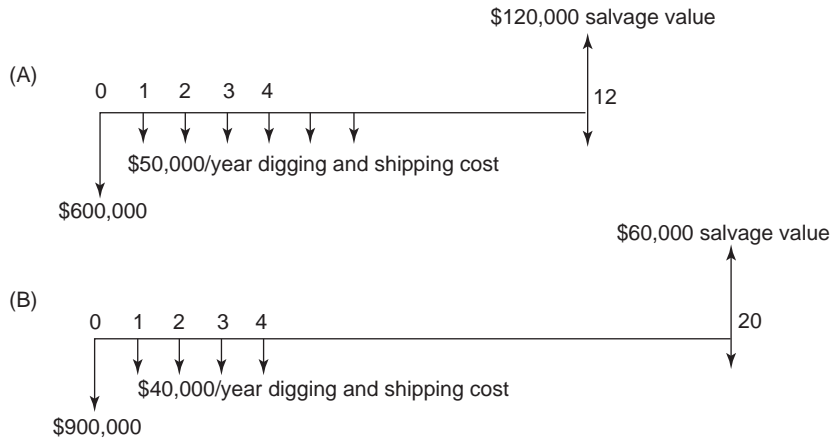


FIGURE A-3 Cash flow diagrams for the mineral quarries of Example A.9. (A) Quarry A. (B) Quarry B.

The cash flow diagrams for the quarries are shown in Fig. A-3. To calculate the annual cost of Quarry A, we need to find the annual cost of an initial investment of \$600,000 using the capital recovery factor (crf), and the equivalent annual income due to salvage of \$120,000 after 12 years. For this income, the sinking fund deposit factor (sdf) will be used. Therefore,

$$\begin{aligned} AC_A &= 600,000 \text{ crf } (0.15, 12) + 50,000 - 120,000 \text{ sdf } (0.15, 12) \\ &= \$156,550.77 \end{aligned}$$

since $\text{crf } (0.15, 12) = 0.18448$ and $\text{sdf } (0.15, 12) = 0.034481$. Similarly,

$$\begin{aligned} AC_B &= 900,000 \text{ crf } (0.15, 20) + 40,000 - 60,000 \text{ sdf } (0.15, 20) \\ &= \$183,199.64 \end{aligned}$$

since $\text{crf } (0.15, 20) = 0.15976$ and $\text{sdf } (0.15, 20) = 0.0097615$. Based on these calculations, Quarry A is a better investment.

A.2.2 Present Worth Comparisons

In present worth (PW) comparisons, all anticipated revenues and expenditures are expressed by their equivalent present values. The same life spans for all the options must be used for valid comparisons. The same sign convention as before shall be used, i.e., a positive sign for costs and a negative sign for receipts. Note also, that in most problems, the present worth of a project is actually its total present cost. Therefore, an alternative with lower present worth is to be preferred. We will solve the examples of the previous subsection again using the present worth method.

EXAMPLE A.10 Alternate Designs

The problem is stated in Example A.7. The cash flow diagrams for the problem are shown in Fig. A-2. We will calculate the present worth of the two designs and compare them. To calculate the present worth of Option A, we need to convert the annual maintenance cost of \$8000 to its present value. For this we use the uniform series present worth factor (uspwf).

Therefore, the present worth of Option A (PW_A) is calculated as

$$\begin{aligned}PW_A &= 280,000 + 8000 \text{ uspwf } (0.12, 50) \\ &= \$346,435.99\end{aligned}$$

since $\text{uspwf } (0.12, 50) = 8.3045$. Similarly, for Option B, we have

$$\begin{aligned}PW_B &= 250,000 + 10,000 \text{ uspwf } (0.12, 50) \\ &= \$333,044.99\end{aligned}$$

Based on these calculations, Option B is a cheaper option. This is the same conclusion as in Example A.7.

EXAMPLE A.11 Alternate Power Stations

The problem is stated in Example A.8. We need to calculate the present worths of the annual cost and the salvage value in order to compare the alternatives by the *present worth method*. We use the uniform series present worth factor (uspwf) to convert the annual cost to its present value. The single payment present worth factor (sppwf) is used to convert the salvage value to its present worth. Therefore,

$$\begin{aligned}PW_A &= 5000 + 2500 \text{ uspwf } (0.15, 4) - 0.3(6000) \text{ sppwf } (0.15, 4) \\ &= \$11,279.82\end{aligned}$$

since $\text{uspwf } (0.15, 4) = 2.8550$ and $\text{sppwf } (0.15, 4) = 0.57175$. Similarly,

$$\begin{aligned}PW_B &= 6000 + 2000 \text{ uspwf } (0.15, 4) - 0.3(6000) \text{ sppwf } (0.15, 4) \\ &= \$10,680.80\end{aligned}$$

$$\begin{aligned}PW_C &= 7500 + 1800 \text{ uspwf } (0.15, 4) - 0.3(7500) \text{ sppwf } (0.15, 4) \\ &= \$11,352.52\end{aligned}$$

Therefore, Bid B is the cheapest option based on these calculations. This is the same conclusion as in Example A.8.

EXAMPLE A.12 Alternate Quarries

The problem is stated in Example A.9. The cash flow diagrams for the two quarries are shown in Fig. A-3. The life span of the two available options is different. We must somehow use the equivalent present worths to compare the alternatives. Several procedures can be used. We shall demonstrate two of them.

Procedure 1. The basic idea here is to calculate the present worth of Quarry B using a 12-year life span, which is the life span of Quarry A. We do this by first calculating the annual cost of Quarry B using a 20-year life span. We then calculate the present worth of the annual cost for the first 12 years only and compare it with the present worth of Quarry A:

$$\begin{aligned}PW_A &= 600,000 + 50,000 \text{ uspwf}(0.15, 12) - 120,000 \text{ sppwf}(0.15, 12) \\ &= \$848,602.09\end{aligned}$$

since $\text{uspwf}(0.15, 12) = 5.42062$ and $\text{sppwf}(0.15, 12) = 0.18691$. Calculating the annual cost of B, we get

$$\begin{aligned}AC_B &= 900,000 \text{ crf}(0.15, 20) + 40,000 - 60,000 \text{ sdf}(0.15, 20) \\ &= \$183,199.64\end{aligned}$$

since $\text{crf}(0.15, 20) = 0.15976$ and $\text{sdf}(0.15, 20) = 0.0097615$. Now, we can calculate the present worth of B using a 12-year life span, as

$$\begin{aligned}PW_B &= 183,199.64 \text{ uspwf}(0.15, 12) \\ &= \$993,055.42\end{aligned}$$

Therefore, Quarry A offers a cheaper solution.

Procedure 2. Since it is known that similar quarries would be available in the near future, we can use 5 A and 3 B quarries to get a total life of 60 years for both options. All future investments, expenses, and salvage values must be converted to their present worth. Using this procedure, the present worth of Quarries A and B are calculated as

$$\begin{aligned}PW_A &= 600,000 + 50,000 \text{ uspwf}(0.15, 60) - 120,000 \text{ sppwf}(0.15, 12) \\ &\quad + 600,000 \text{ uspwf}(0.15, 12) - 120,000 \text{ sppwf}(0.15, 24) \\ &\quad + 600,000 \text{ uspwf}(0.15, 24) - 120,000 \text{ sppwf}(0.15, 36) \\ &\quad + 600,000 \text{ uspwf}(0.15, 36) - 120,000 \text{ sppwf}(0.15, 48) \\ &\quad + 600,000 \text{ uspwf}(0.15, 48) - 120,000 \text{ sppwf}(0.15, 60) \\ &= 600,000 + 333,257.30 + 89,715.43 + 16,768.46 + 3134.14 + 585.79 - 27.37 \\ &= \$1,043,433.7\end{aligned}$$

$$\begin{aligned}PW_B &= 900,000 + 40,000 \text{ uspwf}(0.15, 60) - 60,000 \text{ sppwf}(0.15, 20) \\ &\quad + 900,000 \text{ uspwf}(0.15, 20) - 60,000 \text{ sppwf}(0.15, 40) \\ &\quad + 900,000 \text{ uspwf}(0.15, 40) - 60,000 \text{ sppwf}(0.15, 60) \\ &= 900,000 + 266,605.84 + 51,324.23 + 3135.93 - 13.69 \\ &= \$1,221,052.30\end{aligned}$$

Therefore, with this procedure also, Quarry A is a cheaper option.

Use of the annual base or present worth comparison is dependent on the problem and the available information. For some problems the annual base comparison is better suited while others lend themselves to the present worth method. For each problem a suitable method should be selected.

There are many other factors that must also be considered in the comparison of alternatives. For example, the rate of inflation, the effect of nonuniform payments, and variations in the interest rate should also be considered in comparing alternatives. Most real-world problems will require consideration of such factors. However, these are beyond the scope of the present text. Blank and Tarquin (1983) and other texts on the subject of engineering economy may be consulted for a more comprehensive treatment of these factors.

Exercises for Appendix A

- A.1 A person wants to borrow \$10,000 for one year. The current interest rate is 9 percent compounded monthly. What is the monthly installment to pay back the loan?
- A.2 A person borrows \$15,000 from the bank for one year. The current interest rate is 12 percent compounded daily. If the bank assumes 300 banking days in a year, how much money will have to be paid at the end of the year?
- A.3 A person borrows \$15,000 from the bank for one year. The current interest rate is 12 percent compounded daily. If the bank assumes 365 banking days in a year, how much money will have to be paid at the end of the year?
- A.4 A person promises to pay the bank \$500 every month for the next 24 months. If the interest rate is 9 percent compounded monthly, how much money can the bank lend at the present time?
- A.5 A person deposits \$1000 into the bank at the end of every month. If the prevailing interest rate is 6 percent compounded daily, how much money is accumulated at the end of the year? Assume 30 days in a month.
- A.6 On February 1, 1950 a person went to the bank and promised to pay the bank \$20,000 on February 1, 1951. Based on that promise, how much did the bank lend him at the beginning of every month? The first payment occurred on February 1, 1950 and the last payment on January 1, 1951. Assume the interest rate at that time to be 6 percent compounded monthly.
- A.7 A person decides to deposit \$50 per month for the next 18 years at an annual interest rate of 8 percent compounded monthly. How much money is accumulated at the end of 18 years? How much per month can be withdrawn for the next 4 years after the 18th year?
- A.8 A person borrows \$4000 at an annual interest rate of 13 percent compounded monthly to buy a car and promises to pay it back in 30 monthly installments. What is the monthly installment? How much money would be needed to pay off the loan after the 14th installment?
- A.9 A couple borrows \$200,000 to buy a house. How much is the monthly installment if the interest rate is 8 percent compounded monthly and the money is borrowed for 20 years? If the house is sold for \$250,000 at the beginning of the sixth year (after the 60th installment), how much cash will the owners have after paying back the balance of the loan?

- A.10 A couple deposits \$1000 in a bank at the end of every month for the next 12 months (one year). For the next 12 months (during the second year), they deposit \$1500 every month. If the interest rate is 6 percent compounded monthly, how much money is accumulated in the bank account at the end of two years?
- A.11 A person wants to buy a house costing \$90,000. The prevailing interest rate is 13 percent compounded monthly. The down payment must be 20 percent of the purchase price (i.e., only 80 percent of the purchase price can be borrowed). What is the monthly installment if the loan is for 20 years? How much money will be needed at the end of the 5th year (i.e., at the time of the 60th installment) if the loan has to be paid off at that time?
- A.12 A person invests \$2000 at an annual interest rate of 9 percent compounded monthly. How much money will be received at the end of 2 years? How much money will be received if the interest is compounded daily?
- A.13 A person borrows \$10,000 and agrees to pay \$950 per month for the next 12 months. If the interest is compounded monthly, what is the annual interest rate?
- A.14 On the day a child is born, the parents decide to save a certain amount of money every year for his college education. They decided to deposit the money on every birthday through the 18th starting with the first one, so that the child can withdraw \$10,000 on his 18th, 19th, 20th and 21st birthdays. If the expected rate of return is 9 percent per year, how much money must be deposited annually?
- A.15 A bank can lend \$80,000 for a design project at an annual interest rate of 9 percent compounded monthly. At the time the loan is made, the bank charges 2 percent of the loan as the processing fee. The processing fee can be added to the loan. If the life of the project is 5 years, what is the effective annual interest rate on the \$80,000 loan?
- A.16 A company has two alternative designs for the construction of a bridge. Option A costs \$500,000 initially with an annual maintenance cost of \$10,000. Option B costs \$400,000 initially and its annual maintenance cost is \$12,000. Both options have a salvage value of 5 percent at the end of a 50-year life. Which option should the company adopt? Assume 10 percent per annum as the rate of return. Use both the present worth and annual cost comparison methods.
- A.17 To complete a design project, a person needs \$100,000. Bank A can lend the money at an interest rate of 9 percent compounded annually. Bank B can lend the money at 8 percent compounded annually but it charges 5 percent of the loan as the processing fee. The bank also allows this additional money to be borrowed at an interest rate of 8 percent. Compare the two options using the annual cost comparison method. The life span of the project is 20 years with no salvation value.
- A.18 Compare the two options of Problem A.17 using the present worth method.
- A.19 A county has two options to build a bridge over a creek. Option A calls for a wooden bridge costing \$600,000, and having a life span of 20 years and a maintenance cost of \$10,000 per year. At the end of 10 years, the bridge would require a major renovation costing \$200,000. Option B calls for a concrete and steel bridge costing \$800,000, and having a life span of 40 years and an annual maintenance cost of \$6000. At the end of 20 years this bridge would also need a

major renovation costing \$200,000. There is no salvage value for either the wooden bridge or the concrete bridge at the end of their life. The prevailing interest rate is 8 percent compounded monthly. Compare the two options using the annual cost comparison method.

- A.20 Compare the two options of Problem A.19 using the present worth method.
- A.21 There are two options to complete a project. The life span for the project is 20 years. Option A will cost \$100,000 that needs to be borrowed at an annual interest rate of 9 percent. Option B will cost \$110,000 that will be borrowed at an annual interest rate of 8 percent. In both cases, the interest is compounded annually. Compare the two options using the annual cost method.
- A.22 Compare the two options of Problem A.21 using the present worth method.
- A.23 A city in a mountainous region requires an additional 100-MW peak power capacity and is considering the following alternatives for the next 20-year period, until a breeder reactor meets all needs:
1. Build a new power plant for \$10 million and a \$1 million per year operating cost. The salvage value at the end of 20 years is \$2 million.
 2. Build a pumping/generator station that pumps water to a high lake during periods of low power usage and uses the water for power generation during peak load periods. The initial cost is \$5 million and the operating cost is \$1.5 million per year. There is no salvage value at the end of 20 years.

Which of these alternatives is preferable on a present worth basis? At present interest rates, the following relations hold:

$$S_{20} = 2P, \quad \text{or} \quad P = 0.5S_{20}$$

$$S_{20} = 30R, \quad \text{or} \quad R = \frac{1}{30}S_{20}$$

where S_{20} = value at 20 years, P = present worth, and R = transactions per year.

- A.24 An 80-cm pipeline can be built for \$150,000. The annual operating and maintenance cost is estimated at \$30,000. The alternate 50-cm line can be built for \$120,000. Its operating and maintenance cost is estimated at \$35,000 per year. Either line is expected to serve for 25 years with 10 percent salvage when replaced. Compare the two pipelines on an annual cost and present worth basis assuming a 15 percent rate of return.
- A.25 A company has received two bids for the design and maintenance of a project. Compare the two designs by a present worth analysis using the following data assuming a 10 percent interest rate. Both designs have an economic life of 40 years with no salvage value.

	<i>A</i>	<i>B</i>
First cost (\$)	40,000	50,000
Annual maintenance (\$)	1500	500

- A.26 A person wants to buy a house costing \$100,000. Bank A can lend the money at 12 percent interest compounded monthly. The bank requires 20 percent of the purchase

price as the down payment and charges 2 percent of the loan as a loan processing fee. Bank B also requires a 20 percent down payment. It does not charge a loan processing fee, but its interest rate is 12.5 percent compounded monthly. Both banks require a monthly mortgage payment on the loan and can lend the money for 20 years. Which bank offers the cheaper option to buy the house? Use both the present worth and annual base comparisons.

A.27 Two rental properties are for sale:

	<i>Property I</i>	<i>Property II</i>
Sale price (\$)	600,000	400,000
Annual gross income (\$)	80,000	56,000
Annual management cost (\$)	4,000	3,000
Annual maintenance (\$)	10,000	6,000
Property tax (\$)	12,000	8,000

Each property requires a minimum down payment of 10 percent. The prevailing interest rate is 10 percent compounded annually, and the loan can be obtained for 20 years. Because of the high demand, the value of rental properties is expected to double in 20 years. A person has \$100,000 that can be kept in the bank giving a return of 6 percent or invested into the properties.

- Evaluate the following options using both the annual cost and present worth comparisons assuming a 20-year life for the project:
 - Option A: Buy only Property I (\$100,000 down payment)
 - Option B: Buy only Property II (\$100,000 down payment)
 - Option C: Buy both properties (total down payment \$100,000)
- If the properties are liquidated at the end of 5 years at 110 percent of the purchase price, how much money will be received from Options A, B, and C?

A.28 A company has two options for an operation. The associated costs are:

	<i>First cost</i>	<i>Annual maintenance cost</i>
Option A (\$)	80,000	4,000
Option B (\$)	110,000	2,000

Assume a 40-year life, with zero salvage for either option. Choose the better option, using the present worth comparison with a 15 percent interest rate.

- A.29 To transport its goods, a company has two options: take a slightly longer route to use an existing bridge, or build a new bridge that cuts down the distance and increases the number of trips (which is desirable) for the same cost. The construction of the new bridge will cost \$100,000 and will save \$100 per day over a 240-day working year. The economic life of the bridge is 40 years and the maintenance cost of the structure is estimated at \$1000 per year. Compare the alternatives based on the annual cost and present worth comparisons using a 20 percent annual rate of return.

- A.30 A university is planning to develop a student laboratory. One proposal calls for the construction of the laboratory that would cost \$500,000 and would satisfy the need over the next 12 years. The expected annual operating cost would be \$40,000. After 12 years, an addition to the laboratory would be constructed for \$600,000 with an additional annual operating cost of \$30,000.

The alternative plan is to build a single large laboratory now, which would cost \$650,000. The annual operating cost would be \$42,000 for the first 12 years. At the end of the 12th year, the laboratory would need renovations costing \$100,000 and the annual operating costs would be expected to increase to \$60,000. Compare the two plans using either the annual base or present worth method with an annual interest rate of 10 percent.

- A.31 A company has three options to correct a problem. The costs of the options are (A) \$70,000, (B) \$100,000 and (C) \$50,000. All options are expected to serve for 60 years without any salvage value. However, Option A requires an expense of \$1000 per year, and Option C will require an additional investment of \$80,000 at the end of 20 years, which will have a salvage value of \$15,000 at 60 years from the present time. Which of the three options should be adopted, assuming a 15 percent interest rate?
- A.32 Your company has decided to buy a new company car. You have been designated to evaluate the following three cars and make your recommendation. Assume the following for all cars:

1. The car will be driven 20,000 miles each year for an expected life of 5 years. A total of 90 percent of these will be highway miles; the other 10 percent will be in the city.
2. The price of gasoline is now \$1.25 per gallon and will increase at 10 percent per year for the next 5 years. The gasoline cost will be paid based on the average monthly consumption at the end of each month.
3. At the end of the 5th year, the company will sell the car at a 15 percent salvage value.
4. Insurance will be 5 percent of the original car value per year.
5. The annual rate of return is 8 percent.

The car choices are as follows:

Car A: \$6250 list price

Requires: \$150/year maintenance first 2 years
\$300/year maintenance last 3 years
Mileage estimates: 35 MPG highway
25 MPG city
Financing: \$1000 down payment and the rest in 60 equal monthly payments at 11 percent annual interest

Car B: \$6900 list price

Requires: \$125/year maintenance first 2 years
\$250/year maintenance last 3 years
Mileage estimates: 35 MPG highway
22 MPG city
Financing: \$1500 down payment and the rest in 60 equal monthly payments at 12 percent annual interest

Car C: \$7200 list price

Requires:	\$100/year maintenance first 2 years
	\$200/year maintenance last 3 years
Mileage estimates:	38 MPG highway
	28 MPG city
Financing:	\$1100 down payment and the rest in 60 equal monthly payments at 10 percent annual interest

Use the present worth method of comparison (created by G. Jackson).

Appendix B Vector and Matrix Algebra

Matrix and vector notation is compact and useful in describing many numerical methods and derivations. Matrix and vector algebra is a basic tool needed in developing methods for the optimum design of systems. The solution of linear optimization problems (linear programming) involves an understanding of the solution process for a system of linear equations. Therefore, it is important to understand operations of vector and matrix algebra and be comfortable with their notation. *The subject is often referred to as linear algebra* and has been well-developed for a long time. It has become a standard tool in almost all engineering and scientific applications. In this appendix, some fundamental properties of vectors and matrices are reviewed. For more comprehensive treatment of the subject, several excellent textbooks are available and should be consulted (Hohn, 1964; Franklin, 1968; Cooper and Steinberg, 1970; Stewart, 1973; Bell, 1975; Strang, 1976; Jennings, 1977; Deif, 1982; Gere and Weaver, 1983). In addition, most software libraries have subroutines for linear algebra operations which should be directly utilized.

After reviewing the basic vector and matrix notations, special matrices, determinants, and rank of a matrix, the subject of the solution of a simultaneous system of linear equations is discussed. First an $n \times n$ system and then a rectangular $m \times n$ system are treated. A section on linear independence of vectors is also included. Finally, the eigenvalue problem encountered in many fields of engineering is discussed. Such problems play a prominent role in convex programming problems and sufficiency conditions for optimization.

B.1 Definition of Matrices

A matrix is defined as a rectangular array of quantities that can be real numbers, complex numbers, or functions of several variables. The entries in the rectangular array are also called the *elements of the matrix*. Since the solution of simultaneous linear equations is the most common application of matrices, we use them to develop the notion of matrices.

Consider the following system of two simultaneous linear equations in three unknowns:

$$\begin{aligned}x_1 + 2x_2 + 3x_3 &= 6 \\ -x_1 + 6x_2 - 2x_3 &= 3\end{aligned}\tag{B.1}$$

The symbols x_1, x_2, x_3 represent the *solution variables for the system of equations*. Note that the variables $x_1, x_2,$ and x_3 can be replaced by any other variables, say $w_1, w_2,$ and w_3 , without affecting the solution. Therefore, they are sometimes called the *dummy variables*. Since they are dummy variables, they can be omitted while writing the equations in a matrix form. For example, Eq. (B.1) can be written in a rectangular array as

$$\left[\begin{array}{ccc|c} 1 & 2 & 3 & 6 \\ -1 & 6 & -2 & 3 \end{array} \right]$$

The entries to the left of the vertical line are coefficients of the variables $x_1, x_2,$ and x_3 , and to the right of the vertical line are the numbers on the right side of the equations. It is customary to enclose the array by square brackets as shown. Thus, we see that the system of equations in Eq. (B.1) can be represented by a matrix having two rows and four columns.

An array with m rows and n columns is called a matrix of order “ m by n ,” written as (m, n) or as $m \times n$. To distinguish between matrices and scalars, we shall boldface the variables that represent matrices. In addition, capital letters will be used to represent matrices. For example, a general matrix \mathbf{A} of order $m \times n$ can be represented as

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \quad (\text{B.2})$$

The coefficients a_{ij} are called elements of the matrix \mathbf{A} ; subscripts i and j indicate the row and column numbers for the element a_{ij} (e.g., a_{32} represents the element in the third row and second column). Although the elements can be real numbers, complex numbers, or functions, we shall not deal with complex matrices in the present text. We shall encounter matrices having elements as functions of several variables, e.g., the Hessian matrix of a function discussed in Chapter 4.

It is useful to employ more compact notation for matrices. For example, a matrix \mathbf{A} of order $m \times n$ with a_{ij} 's as its elements is written compactly as

$$\mathbf{A} = [a_{ij}]_{(m \times n)} \quad (\text{B.3})$$

Often, the size of the matrix is also not shown and \mathbf{A} is written as $[a_{ij}]$.

If a matrix has the same number of rows and columns, then it is called a *square matrix*. In Eq. (B.2) or (B.3), if $m = n$, \mathbf{A} is a square matrix. It is called a matrix of order n .

It is important to understand the matrix notation for a set of linear equations because we shall encounter such equations quite often in this text. For example, Eq. (B.1) can be written as

$$\begin{array}{ccc} x_1 & x_2 & x_3 & \mathbf{b} \\ \left[\begin{array}{ccc|c} 1 & 2 & 3 & 6 \\ -1 & 6 & -2 & 3 \end{array} \right] \end{array}$$

The preceding array containing coefficients of the equations and the right-side parameters is called the *augmented matrix*. Note that *each column of the matrix is identified with a variable*; the first column is associated with the variable x_1 because it contains coefficients of x_1 for all equations, the second with x_2 , the third with x_3 , and the last column with the right-side

vector, which we call \mathbf{b} . This interpretation is important while solving linear simultaneous equations (discussed later) or linear programming problems (discussed in Chapter 6).

B.2 Type of Matrices and Their Operations

B.2.1 Null Matrix

A matrix having all zero elements is called a *null (zero) matrix* denoted by boldfaced zero as $\mathbf{0}$. Any *zero matrix* of proper order when premultiplied or postmultiplied by any other matrix (or scalar) results in a zero matrix.

B.2.2 Vector

A matrix of order $1 \times n$ is called a *row matrix*, or simply *row vector*. Similarly, a matrix of order $n \times 1$ is called a *column matrix*, or simply *column vector*. A vector with n elements is called an n -component vector, or an n -vector. In this text, all vectors are considered to be column vectors and denoted by a lower-case letter in boldface.

B.2.3 Addition of Matrices

If \mathbf{A} and \mathbf{B} are two matrices of the order $m \times n$, then their *sum* is also an $m \times n$ matrix defined as

$$\mathbf{C}_{(m \times n)} = \mathbf{A} + \mathbf{B}; \quad c_{ij} = a_{ij} + b_{ij} \text{ for all } i \text{ and } j \quad (\text{B.4})$$

Matrix addition satisfies the following properties

$$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A} \quad (\text{commutative}) \quad (\text{B.5})$$

If \mathbf{A} , \mathbf{B} , and \mathbf{C} are three matrices of the same order, then

$$\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C} \quad (\text{associative}) \quad (\text{B.6})$$

If \mathbf{A} , \mathbf{B} , and \mathbf{C} are of same order, then

$$\mathbf{A} + \mathbf{C} = \mathbf{B} + \mathbf{C} \text{ implies } \mathbf{A} = \mathbf{B} \quad (\text{B.7})$$

where $\mathbf{A} = \mathbf{B}$ implies that the matrices are equal. Two matrices \mathbf{A} and \mathbf{B} of order $m \times n$ are equal if $a_{ij} = b_{ij}$ for $i = 1$ to m and $j = 1$ to n .

B.2.4 Multiplication of Matrices

Multiplication of a matrix \mathbf{A} of order $m \times n$ by a scalar k is defined as

$$k\mathbf{A} = [ka_{ij}]_{(m \times n)} \quad (\text{B.8})$$

The multiplication (product) \mathbf{AB} of two matrices \mathbf{A} and \mathbf{B} is defined only if \mathbf{A} and \mathbf{B} are of proper order. The number of columns of \mathbf{A} must be equal to the number of rows of \mathbf{B} for the product \mathbf{AB} to be defined. In that case, the matrices are said to be *conformable* for multiplication. If \mathbf{A} is $m \times n$ and \mathbf{B} is $r \times p$, then the multiplication \mathbf{AB} is defined only when $n = r$, and multiplication \mathbf{BA} is defined only when $m = p$. Multiplication of two matrices of proper order results in a third matrix. If \mathbf{A} and \mathbf{B} are of order $m \times n$ and $n \times p$ respectively, then

$$\mathbf{AB} = \mathbf{C} \quad (\text{B.9a})$$

where \mathbf{C} is a matrix of order $m \times p$. Elements of the matrix \mathbf{C} are determined by multiplying the elements of a row of \mathbf{A} with the elements of a column of \mathbf{B} and adding all the multiplications. Thus

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \vdots & \vdots & & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mp} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1p} \\ c_{21} & c_{22} & \dots & c_{2p} \\ \vdots & \vdots & & \vdots \\ c_{m1} & c_{m2} & \dots & c_{mp} \end{bmatrix} \quad (\text{B.9b})$$

where elements c_{ij} are calculated as

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj} = \sum_{k=1}^n a_{ik}b_{kj} \quad (\text{B.10})$$

Note that if \mathbf{B} is an $n \times 1$ matrix (i.e., a vector), then \mathbf{C} is an $m \times 1$ matrix. We shall encounter this type of matrix multiplication quite often in this text, e.g., a linear system of equations is represented as $\mathbf{Ax} = \mathbf{b}$, where \mathbf{x} contains the solution variables and \mathbf{b} the right-side parameters. Equation (B.1) can be written in this form.

In the product \mathbf{AB} the matrix \mathbf{A} is said to be *postmultiplied* by \mathbf{B} or \mathbf{B} is said to be *pre-multiplied* by \mathbf{A} . Whereas the matrix addition satisfies commutative law, matrix multiplication does not, in general, satisfy this law, i.e., $\mathbf{AB} \neq \mathbf{BA}$. Also, even if \mathbf{AB} is well defined, \mathbf{BA} may not be defined.

EXAMPLE B.1 Multiplication of Matrices

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 1 \\ 6 & 3 & 2 \\ 4 & 2 & 0 \\ 0 & 3 & 5 \end{bmatrix}_{(4 \times 3)} \quad \mathbf{B} = \begin{bmatrix} 2 & -1 \\ 1 & 0 \\ 3 & -2 \end{bmatrix}_{(3 \times 2)} \quad (\text{a})$$

$$\mathbf{AB} = \begin{bmatrix} 2 & 3 & 1 \\ 6 & 3 & 2 \\ 4 & 2 & 0 \\ 0 & 3 & 5 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ 1 & 0 \\ 3 & -2 \end{bmatrix} \quad (\text{b})$$

$$= \begin{bmatrix} (2 \times 2 + 3 \times 1 + 1 \times 3) & (-2 \times 1 + 3 \times 0 - 1 \times 2) \\ (6 \times 2 + 3 \times 1 + 2 \times 3) & (-6 \times 1 + 3 \times 0 - 2 \times 2) \\ (4 \times 2 + 2 \times 1 + 0 \times 3) & (-4 \times 1 + 2 \times 0 - 0 \times 2) \\ (0 \times 2 + 3 \times 1 + 5 \times 3) & (-0 \times 1 + 3 \times 0 - 5 \times 2) \end{bmatrix} \quad (\text{c})$$

$$= \begin{bmatrix} 10 & -4 \\ 21 & -10 \\ 10 & -4 \\ 18 & -10 \end{bmatrix}_{(4 \times 2)} \quad (\text{d})$$

Note that the product \mathbf{BA} is not defined because the number of columns in \mathbf{B} is not equal to the number of rows in \mathbf{A} .

EXAMPLE B.2 Multiplication of Matrices

$$\mathbf{A} = \begin{bmatrix} 4 & 1 \\ -2 & 0 \\ 8 & 3 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} -3 & 8 & 6 \\ 8 & 3 & -1 \end{bmatrix} \quad (\text{a})$$

$$\mathbf{AB} = \begin{bmatrix} 4 & 1 \\ -2 & 0 \\ 8 & 3 \end{bmatrix} \begin{bmatrix} -3 & 8 & 6 \\ 8 & 3 & -1 \end{bmatrix} = \begin{bmatrix} -4 & 35 & 23 \\ 6 & -16 & -12 \\ 0 & 73 & 45 \end{bmatrix} \quad (\text{b})$$

$$\mathbf{BA} = \begin{bmatrix} -3 & 8 & 6 \\ 8 & 3 & -1 \end{bmatrix} \begin{bmatrix} 4 & 1 \\ -2 & 0 \\ 8 & 3 \end{bmatrix} = \begin{bmatrix} 20 & 15 \\ 18 & 5 \end{bmatrix} \quad (\text{c})$$

Note that for the products \mathbf{AB} and \mathbf{BA} to be matrices of the same order, \mathbf{A} and \mathbf{B} must be square matrices.

Note that even if matrices \mathbf{A} , \mathbf{B} , and \mathbf{C} are properly defined, $\mathbf{AB} = \mathbf{AC}$ does not imply $\mathbf{B} = \mathbf{C}$. Also, if $\mathbf{AB} = \mathbf{0}$, it does not imply either $\mathbf{B} = \mathbf{0}$ or $\mathbf{A} = \mathbf{0}$. The matrix multiplication, however, satisfies two important laws: the associative and distributive laws. Let matrices \mathbf{A} , \mathbf{B} , \mathbf{C} , \mathbf{D} , and \mathbf{F} be of proper dimension. Then

$$1. (\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC}) \quad (\text{associative law}) \quad (\text{B.11})$$

$$2\text{a. } \mathbf{B}(\mathbf{C} + \mathbf{D}) = \mathbf{BC} + \mathbf{BD} \quad (\text{distributive laws})$$

$$2\text{b. } (\mathbf{C} + \mathbf{D})\mathbf{F} = \mathbf{CF} + \mathbf{DF} \quad (\text{B.12})$$

$$2\text{c. } (\mathbf{A} + \mathbf{B})(\mathbf{C} + \mathbf{D}) = \mathbf{AC} + \mathbf{AD} + \mathbf{BC} + \mathbf{BD}$$

B.2.5 Transpose of a Matrix

We can write rows of a matrix as columns and obtain another matrix. Such an operation is called the transpose of a matrix. If $\mathbf{A} = [a_{ij}]$ is an $m \times n$ matrix, then its transpose, denoted as \mathbf{A}^T , is an $n \times m$ matrix. It is obtained from \mathbf{A} by interchanging its rows and columns. The first column of \mathbf{A} is the first row of \mathbf{A}^T ; the second column of \mathbf{A} is the second row of \mathbf{A}^T ; and so on. Thus, if $\mathbf{A} = [a_{ij}]$, then $\mathbf{A}^T = [a_{ji}]$. The operation of transposing a matrix is illustrated by the following 2×3 matrix:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}; \quad \mathbf{A}^T = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{bmatrix}$$

Some properties of the transpose are

1. $(\mathbf{A}^T)^T = \mathbf{A}$
2. $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$
3. $(\alpha\mathbf{A})^T = \alpha\mathbf{A}^T$, $\alpha = \text{scalar}$
4. $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$

B.2.6 Elementary Row–Column Operations

There are three simple but extremely useful operations for rows or columns of a matrix. They are used in later discussions, so we state them here:

1. Interchange any two rows (columns).
2. Multiply any row (column) by a nonzero scalar.
3. Add to any row (column) a scalar multiple of another row (column).

B.2.7 Equivalence of Matrices

A matrix \mathbf{A} is said to be *equivalent* to another matrix \mathbf{B} written as $\mathbf{A} \sim \mathbf{B}$ if \mathbf{A} can be transformed into \mathbf{B} by means of one or more elementary row and/or column operations. If only row (column) operations are used, we say \mathbf{A} is *row (column) equivalent* to \mathbf{B} .

B.2.8 Scalar Product–Dot Product of Vectors

A special case of matrix multiplication of particular interest is the multiplication of a row vector by a column vector. If \mathbf{x} and \mathbf{y} are two n -component vectors, then

$$\mathbf{x}^T \mathbf{y} = \sum_{j=1}^n x_j y_j \quad (\text{B.13})$$

where

$$\mathbf{x}^T = [x_1 \quad x_2 \quad \dots \quad x_n] \quad \text{and} \quad \mathbf{y} = [y_1 \quad y_2 \quad \dots \quad y_n]^T$$

The product in Eq. (B.13) is called the *scalar product or dot product* of \mathbf{x} and \mathbf{y} . It is also denoted as $\mathbf{x} \cdot \mathbf{y}$. Note that since the dot product of two vectors is a scalar, $\mathbf{x}^T \mathbf{y} = \mathbf{y}^T \mathbf{x}$.

Associated with any vector \mathbf{x} is a scalar called *norm or length of the vector* defined as

$$(\mathbf{x}^T \mathbf{x})^{1/2} = \left(\sum_{i=1}^n x_i^2 \right)^{1/2} \quad (\text{B.14})$$

Often the norm of \mathbf{x} is designated as $\|\mathbf{x}\|$.

B.2.9 Square Matrices

A matrix having the same number of rows and columns is called a *square matrix*; otherwise it is called a rectangular matrix. The elements a_{ii} , $i = 1$ to n are called the *main diagonal elements* and others are called the *off-diagonal elements*. A square matrix having zero entries at all off-diagonal locations is called a *diagonal matrix*. If all main diagonal elements of a diagonal matrix are equal, it is called a *scalar matrix*.

A square matrix \mathbf{A} is called *symmetric* if $\mathbf{A}^T = \mathbf{A}$ and *asymmetric or unsymmetric* otherwise. It is called *antisymmetric* if $\mathbf{A}^T = -\mathbf{A}$.

If all the elements below the main diagonal of a square matrix are zero ($a_{ij} = 0$ for $i > j$), it is called an *upper triangular matrix*. Similarly, a *lower triangular matrix* has all zero elements above the main diagonal ($a_{ij} = 0$ for $i < j$). A matrix that has all zero entries except in a band around the main diagonal is called a *banded matrix*.

A square matrix having unit elements on the main diagonal and zeros elsewhere is called an *identity matrix*. An identity matrix of order n is denoted as $\mathbf{I}_{(n)}$. Identity matrices are useful because their pre- or postmultiplication with another matrix does not change the matrix, e.g., let \mathbf{A} be any $m \times n$ matrix, then

$$\mathbf{I}_{(m)}\mathbf{A} = \mathbf{A} = \mathbf{A}\mathbf{I}_{(n)} \quad (\text{B.15})$$

A scalar matrix $\mathbf{S}_{(n)}$ having diagonal elements as α can be written as

$$\mathbf{S}_{(n)} = \alpha\mathbf{I}_{(n)} \quad (\text{B.16})$$

Note that premultiplying or postmultiplying any matrix by a scalar matrix of proper order results in multiplying the original matrix by the scalar. This can be proved for any $m \times n$ matrix \mathbf{A} as follows:

$$\mathbf{S}_{(m)}\mathbf{A} = \alpha\mathbf{I}_{(m)}\mathbf{A} = \alpha\mathbf{A} = \alpha(\mathbf{A}\mathbf{I}_{(n)}) = \mathbf{A}(\alpha\mathbf{I}_{(n)}) = \mathbf{A}\mathbf{S}_{(n)} \quad (\text{B.17})$$

B.2.10 Partitioning of Matrices

It is often useful to divide vectors and matrices into a smaller group of elements. This can be done by partitioning the matrix into smaller rectangular arrays called *submatrices* and a vector into *subvectors*. For example, consider a matrix \mathbf{A} as

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & -6 & 4 & 3 \\ 2 & 3 & 8 & -1 & -3 \\ 1 & -6 & 2 & 3 & 8 \\ -3 & 0 & 5 & -2 & 7 \end{bmatrix}_{(4 \times 5)}$$

A possible partitioning of \mathbf{A} is

$$\mathbf{A} = \left[\begin{array}{ccc|cc} 2 & 1 & -6 & 4 & 3 \\ 2 & 3 & 8 & -1 & -3 \\ \hline 1 & -6 & 2 & 3 & 8 \\ -3 & 0 & 5 & -2 & 7 \end{array} \right]_{(4 \times 5)}$$

Therefore, submatrices of \mathbf{A} are

$$\begin{aligned} \mathbf{A}_{11} &= \begin{bmatrix} 2 & 1 & -6 \\ 2 & 3 & 8 \end{bmatrix}_{(2 \times 3)} & \mathbf{A}_{12} &= \begin{bmatrix} 4 & 3 \\ -1 & -3 \end{bmatrix}_{(2 \times 2)} \\ \mathbf{A}_{21} &= \begin{bmatrix} 1 & -6 & 2 \\ -3 & 0 & 5 \end{bmatrix}_{(2 \times 3)} & \mathbf{A}_{22} &= \begin{bmatrix} 3 & 8 \\ -2 & 7 \end{bmatrix}_{(2 \times 2)} \end{aligned}$$

where \mathbf{A}_{ij} are matrices of proper order. Thus, \mathbf{A} can be written in terms of submatrices as

$$\mathbf{A} = \left[\begin{array}{cc|cc} \mathbf{A}_{11} & \mathbf{A}_{12} & & \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} & & \end{array} \right]$$

Note that partitioning of vectors and matrices must be proper so that the operations of addition or multiplication remain defined. To see how two partitioned matrices are multiplied, consider \mathbf{A} an $m \times n$ and \mathbf{B} an $n \times p$ matrix. Let these be partitioned as

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} \end{array} \right]_{(m \times n)} ; \quad \mathbf{B} = \left[\begin{array}{c|c} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \hline \mathbf{B}_{21} & \mathbf{B}_{22} \end{array} \right]_{(n \times p)}$$

Then the product \mathbf{AB} can be written as

$$\mathbf{AB} = \left[\begin{array}{c|c} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \hline \mathbf{A}_{21} & \mathbf{A}_{22} \end{array} \right] \left[\begin{array}{c|c} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \hline \mathbf{B}_{21} & \mathbf{B}_{22} \end{array} \right] = \left[\begin{array}{c|c} (\mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21}) & (\mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22}) \\ \hline (\mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21}) & (\mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22}) \end{array} \right]_{(m \times p)}$$

Note that the partitioning of matrices \mathbf{A} and \mathbf{B} must be such that the matrix products $\mathbf{A}_{11}\mathbf{B}_{11}$, $\mathbf{A}_{12}\mathbf{B}_{21}$, $\mathbf{A}_{11}\mathbf{B}_{12}$, $\mathbf{A}_{12}\mathbf{B}_{22}$, etc., are proper. In addition, the pairs of matrices $\mathbf{A}_{11}\mathbf{B}_{11}$ and $\mathbf{A}_{12}\mathbf{B}_{21}$, $\mathbf{A}_{11}\mathbf{B}_{12}$ and $\mathbf{A}_{12}\mathbf{B}_{22}$, etc., must be of the same order.

B.3 Solution of n Linear Equations in n Unknowns

B.3.1 Linear Systems

Linear systems of equations are encountered in numerous engineering and scientific applications. Therefore, substantial research and development work has been done to devise several solution procedures. It is critically important to understand the basic ideas and concepts related to linear equations because we use them quite often in this text. In this section we shall describe a basic procedure, known as *Gaussian elimination*, for solution of an $n \times n$ (square) linear system of equations. More general methods for solving a rectangular $m \times n$ system are discussed in the next section.

It turns out that the idea of determinants is closely related to the solution of a linear system of equations so first we discuss determinants and their properties. It also turns out that the solution of a square system can be found by inverting the matrix associated with the system so we describe methods for inverting matrices.

Let us consider the following system of n equations in n unknowns:

$$\mathbf{Ax} = \mathbf{b} \tag{B.18}$$

where \mathbf{A} is an $n \times n$ matrix of specified constants, \mathbf{x} is an n -vector of solution variables, and \mathbf{b} is an n -vector of specified constants known as the right-side vector. \mathbf{A} is called the *coefficient matrix* and when the vector \mathbf{b} is added as the $(n + 1)$ th column of \mathbf{A} as $[\mathbf{A} \mid \mathbf{b}]$, the resulting matrix is called the *augmented matrix* for the given system of equations. Note that the left side of Eq. (B.18) consists of multiplication of an $n \times n$ matrix with an n -component vector resulting in another n -component vector. If the right-side vector \mathbf{b} is zero, Eq. (B.18) is called a *homogeneous* system; otherwise it is called the *nonhomogeneous system* of equations.

The equation $\mathbf{Ax} = \mathbf{b}$ can also be written in the following summation form:

$$\sum_{j=1}^n a_{ij}x_j = b_i; \quad i = 1 \text{ to } n \tag{B.19}$$

If each row of the matrix \mathbf{A} is interpreted as an n -dimensional row vector $\bar{\mathbf{a}}^{(i)}$, then the left side of Eq. (B.19) can be interpreted as the dot product of two vectors as

$$(\bar{\mathbf{a}}^{(i)} \cdot \mathbf{x}) = b_i; \quad i = 1 \text{ to } n \tag{B.20}$$

If each column of \mathbf{A} is interpreted as an n -dimensional column vector $\mathbf{a}^{(i)}$, then the left side of Eq. (B.18) can be interpreted as the summation of the scaled columns of the matrix \mathbf{A} as

$$\sum_{i=1}^n \mathbf{a}^{(i)} x_i = \mathbf{b} \quad (\text{B.21})$$

These interpretations can be useful in devising solution strategies for the system $\mathbf{Ax} = \mathbf{b}$ and in their implementation. For example, Eq. (B.21) shows that the solution variable x_i is simply a scale factor for the i th column of \mathbf{A} , i.e., variable x_i is associated with the i th column.

B.3.2 Determinants

To develop the solution strategies for the linear system $\mathbf{Ax} = \mathbf{b}$, we begin by introducing the concept of determinants and study their properties. The methods for calculating determinants are intimately related to the procedures for solving linear equations, so we shall also discuss them.

Every square matrix has a scalar associated with it, called the determinant calculated from its elements. To introduce the idea of determinants, we set $n = 2$ in Eq. (B.18) and consider the following 2×2 system of simultaneous equations:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad (\text{a})$$

The number $(a_{11}a_{22} - a_{21}a_{12})$ calculated using elements of the coefficient matrix is called its determinant. To see how this number arises, we shall solve the system in Eq. (a) by the *elimination process*.

Multiplying the first row by a_{22} and the second by a_{12} in Eq. (a), we get

$$\begin{bmatrix} a_{11}a_{22} & a_{12}a_{22} \\ a_{12}a_{21} & a_{12}a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{22}b_1 \\ a_{12}b_2 \end{bmatrix} \quad (\text{b})$$

Subtracting the second row from the first one in Eq. (b), we eliminate x_2 from the first equation and obtain:

$$(a_{11}a_{22} - a_{12}a_{21})x_1 = a_{22}b_1 - a_{12}b_2 \quad (\text{c})$$

Now repeating the foregoing process to eliminate x_1 from the second row in Eq. (a) by multiplying the first equation by a_{21} and the second by a_{11} and subtracting, we obtain:

$$(a_{11}a_{22} - a_{12}a_{21})x_2 = a_{11}b_2 - a_{21}b_1 \quad (\text{d})$$

The coefficient of x_1 and x_2 in Eqs. (c) and (d) must be nonzero for a unique solution of the system, i.e., $(a_{11}a_{22} - a_{12}a_{21}) \neq 0$, and the values of x_1 and x_2 are calculated as

$$x_1 = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{12}a_{21}}, \quad x_2 = \frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{12}a_{21}} \quad (\text{e})$$

The denominator $(a_{11}a_{22} - a_{12}a_{21})$ is identified as the *determinant* of the matrix \mathbf{A} of Eq. (a). It is denoted by $\det(\mathbf{A})$, or $|\mathbf{A}|$. Thus, for any 2×2 matrix \mathbf{A} ,

$$|\mathbf{A}| = a_{11}a_{22} - a_{12}a_{21} \quad (\text{f})$$

Using the definition of Eq. (f), we can rewrite Eq. (e) as

$$x_1 = \frac{|\mathbf{B}_1|}{|\mathbf{A}|}, \quad x_2 = \frac{|\mathbf{B}_2|}{|\mathbf{A}|} \quad (\text{g})$$

where \mathbf{B}_1 is obtained by replacing the first column of \mathbf{A} with the right side and \mathbf{B}_2 is obtained by replacing the second column of \mathbf{A} with the right side, as

$$\mathbf{B}_1 = \begin{bmatrix} b_1 & a_{12} \\ b_2 & a_{22} \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} a_{11} & b_1 \\ a_{21} & b_2 \end{bmatrix} \quad (\text{h})$$

Equation (g) is known as *Cramer's rule*. According to this rule, we need to compute only the three determinants— $|\mathbf{A}|$, $|\mathbf{B}_1|$, and $|\mathbf{B}_2|$ —to determine the solution to any 2×2 system of equations. If $|\mathbf{A}| = 0$, there is no unique solution to Eq. (a). There may be an infinite number of solutions or no solution at all. These cases are investigated in the next section.

The preceding concept of a determinant can be generalized to $n \times n$ matrices. For every square matrix \mathbf{A} of any order, we can associate a unique scalar, called the determinant of \mathbf{A} . There are many ways of calculating the determinant of a matrix. These procedures are closely related to the ones used for solving the linear system of equations that we shall discuss later in this section.

Properties of Determinants The determinants have several properties that are useful in devising procedures for their calculation. Therefore, these should be clearly understood.

1. The determinant of any square matrix \mathbf{A} is also equal to the determinant of the transpose of the matrix, i.e., $|\mathbf{A}| = |\mathbf{A}^T|$.
2. If a square matrix \mathbf{A} has two identical columns (or rows), then its determinant is zero, i.e., $|\mathbf{A}| = 0$.
3. If a new matrix is formed by interchanging any two columns (or rows) of a given matrix \mathbf{A} (elementary row–column Operation 1), the determinant of the resulting matrix is the negative of the determinant of the original matrix.
4. If a new matrix is formed by adding any multiple of one column (row) to a different column (row) of a given matrix (elementary row–column Operation 3), the determinant of the resulting matrix is equal to the determinant of the original matrix.
5. If a square matrix \mathbf{B} is identical to a matrix \mathbf{A} , except some column (or row) is a scalar multiple c of the corresponding column (or row) of \mathbf{A} (elementary row–column Operation 2), then $|\mathbf{B}| = c |\mathbf{A}|$.
6. If elements of a column (or row) of a square matrix \mathbf{A} are zero, then $|\mathbf{A}| = 0$.
7. If a square matrix \mathbf{A} is lower or upper triangular, then the determinant of \mathbf{A} is equal to the product of the diagonal elements:

$$|\mathbf{A}| = a_{11}a_{22} \cdots a_{nn} \quad (\text{B.22})$$

8. If \mathbf{A} and \mathbf{B} are any two square matrices of the same order, then

$$|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}|$$

9. Let $|\mathbf{A}_{ij}|$ denote the determinant of a matrix obtained by deleting the i th row and the j th column of \mathbf{A} (yielding a square matrix of order $n - 1$); the scalar $|\mathbf{A}_{ij}|$ is called the *minor* of the element a_{ij} of matrix \mathbf{A} . Then the *cofactor* of a_{ij} is defined as

$$\text{cofac}(a_{ij}) = (-1)^{i+j} |\mathbf{A}_{ij}| \quad (\text{B.23})$$

The determinant of \mathbf{A} is calculated in terms of the cofactors as

$$|\mathbf{A}| = \sum_{j=1}^n a_{ij} \text{cofac}(a_{ij}), \quad \text{for any } i \quad (\text{B.24})$$

Or,

$$|\mathbf{A}| = \sum_{i=1}^n a_{ij} \text{cofac}(a_{ij}), \quad \text{for any } j \quad (\text{B.25})$$

Note that the $\text{cofac}(a_{ij})$ is also a scalar obtained from the minor $|\mathbf{A}_{ij}|$ but having a positive or negative sign determined by the indices i and j as $(-1)^{i+j}$. Equation (B.24) is called the cofactor expansion for $|\mathbf{A}|$ by the i th row; Eq. (B.25) is called the cofactor expansion for $|\mathbf{A}|$ by the j th column. Equations (B.24) and (B.25) can be used to prove Properties 2, 5, 6, and 7 directly.

It is important to note that Eq. (B.24) or (B.25) is difficult to use to calculate the determinant of \mathbf{A} . These equations require calculation of the cofactors of the elements a_{ij} , which are determinants in themselves. However, using the *elementary row and column operations*, a square matrix can be converted to either lower or upper triangular form. The determinant is then computed using Eq. (B.22). This will be illustrated in an example later in this section.

A matrix having a zero determinant is called a singular matrix; a matrix with a nonzero determinant is called nonsingular. A nonhomogeneous $n \times n$ system of equations has a unique solution if and only if the matrix of coefficients is nonsingular. These properties are discussed and used subsequently to develop methods for solving a system of equations.

Leading Principal Minor Every $n \times n$ square matrix \mathbf{A} has certain scalars associated with it, called the leading principal minors. They are obtained as determinants of certain submatrices of \mathbf{A} . They are useful in determining the “form” of a matrix that is needed in checking sufficiency conditions for optimality as well as the convexity of functions discussed in Chapter 4. Therefore, we discuss the idea of leading principal minors here.

Let M_k , $k = 1$ to n be called the leading principal minors of \mathbf{A} . Then each M_k is defined as the determinant of the following submatrix:

$$M_k = |\mathbf{A}_{kk}| \quad (\text{B.26})$$

where \mathbf{A}_{kk} is a $k \times k$ submatrix of \mathbf{A} obtained by deleting the last $(n - k)$ columns and the corresponding rows. For example, $M_1 = a_{11}$, $M_2 =$ determinant of a 2×2 matrix obtained by deleting all rows and columns of \mathbf{A} except the first two, and so on, are the leading principal minors of the matrix \mathbf{A} .

B.3.3 Gaussian Elimination Procedure

The elimination process described earlier in Section B.3.1 for solving a 2×2 system of equations can be generalized to solve any $n \times n$ system of equations. The entire process can be

organized and explained using the matrix notation. The procedure can also be used to calculate the determinant of any matrix. The procedure is known as *Gaussian elimination*, which we shall describe in detail in the following.

Using the three elementary row–column operations defined in Section B.2, the system $\mathbf{Ax} = \mathbf{b}$ of Eq. (B.18) can be transformed to the following form:

$$\begin{bmatrix} 1 & \bar{a}_{12} & \bar{a}_{13} & \dots & \bar{a}_{1n} \\ 0 & 1 & \bar{a}_{23} & \dots & \bar{a}_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \bar{b}_1 \\ \bar{b}_2 \\ \vdots \\ \bar{b}_n \end{bmatrix} \quad (\text{B.27})$$

Or, in expanded form, Eq. (B.27) becomes

$$\begin{aligned} x_1 + \bar{a}_{12}x_2 + \bar{a}_{13}x_3 + \dots + \bar{a}_{1n}x_n &= \bar{b}_1 \\ x_2 + \bar{a}_{23}x_3 + \dots + \bar{a}_{2n}x_n &= \bar{b}_2 \\ x_3 + \dots + \bar{a}_{3n}x_n &= \bar{b}_3 \\ &\vdots \\ x_n &= \bar{b}_n \end{aligned} \quad (\text{B.28})$$

Note that we use \bar{a}_{ij} and \bar{b}_i to represent modified elements a_{ij} and b_i of the original system. From the n th equation of the system (B.28), we have $x_n = \bar{b}_n$. If we substitute this value into the $(n - 1)$ th equation of (B.28), we can solve for x_{n-1} :

$$\begin{aligned} x_{n-1} &= \bar{b}_{n-1} - \bar{a}_{n-1,n}x_n \\ &= \bar{b}_{n-1} - \bar{a}_{n-1,n}\bar{b}_n \end{aligned} \quad (\text{B.29})$$

Equation (B.29) can now be substituted into the $(n - 2)$ th equation of (B.28) and x_{n-2} can be determined. Continuing in this manner, each of the unknowns can be solved in reverse order: $x_n, x_{n-1}, x_{n-2}, \dots, x_2, x_1$. The procedure of reducing a system of n equations in n unknowns and then solving successively for $x_n, x_{n-1}, x_{n-2}, \dots, x_2, x_1$ is called the *Gaussian elimination procedure* or *Gauss reduction*. The latter part of the method (solving successively for $x_n, x_{n-1}, x_{n-2}, \dots, x_2, x_1$) is called the *backward substitution*, or *backward pass*.

The *Gaussian elimination procedure* uses elementary row–column operations to convert the main diagonal elements of the given coefficient matrix to 1 and the elements below the main diagonal to zero. To carry out these operations we start with the first row and the first column of the given matrix augmented with the right side of the system of equations. To make the diagonal element 1, the first row is divided by the diagonal element. To convert the elements in the first column below the main diagonal to zero, we multiply the first row by the element \bar{a}_{i1} in the i th row ($i = 2$ to n). The resulting elements of the first row are subtracted from the i th row. This makes the element \bar{a}_{i1} zero in the i th row. These operations are carried out for each row using the first row for elimination each time. Once all the elements below the main diagonal are zero in the first column, the procedure is repeated for the second column using the second row for elimination, and so on. The row used to obtain zero elements in a column is called the *pivot row*, and the column in which elimination is performed is called the *pivot column*. We will illustrate this procedure in an example later.

The foregoing operations of converting elements below the main diagonal to zero can be explained in another way. When we make the elements below the main diagonal in the first column zero, we are eliminating the variable x_1 from all the equations except the first equa-

tion (x_1 is associated with the first column). For this elimination step we use the first equation. In general, when we reduce the elements below the main diagonal in the i th column to zero, we use the i th row as the pivot row. Thus, we eliminate the i th variable from all the equations below the i th row. This explanation is quite straightforward once we realize that each column of the coefficient matrix has a variable associated with it, as noted before.

EXAMPLE B.3 Solution of Equations by Gaussian Elimination

Solve the following 3×3 system of equations

$$\begin{aligned}x_1 - x_2 + x_3 &= 0 \\x_1 - x_2 + 2x_3 &= 1 \\x_1 + x_2 + 2x_3 &= 5\end{aligned}\tag{a}$$

Solution. We shall illustrate the Gaussian elimination procedure in a step-by-step manner using the augmented matrix idea. The augmented matrix for Eq. (a) is defined using the coefficients of the variables in each equation and the right-side parameters, as

$$\mathbf{B} = \begin{array}{ccc|c} x_1 & x_2 & x_3 & \mathbf{b} \\ \hline 1 & -1 & 1 & 0 \\ 1 & -1 & 2 & 1 \\ 1 & 1 & 2 & 5 \end{array}\tag{b}$$

To convert the preceding system to the form of Eq. (B.27), we use the elementary row-column operations as follows:

1. Add -1 times row 1 to row 2 and -1 times row 1 to row 3 (eliminating x_1 from the second and third equations; elementary row operation 3):

$$\mathbf{B} \sim \begin{array}{ccc|c} x_1 & x_2 & x_3 & \mathbf{b} \\ \hline 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 1 & 5 \end{array} \quad \begin{array}{l} \text{(recall that symbol “}\sim\text{”} \\ \text{means equivalence} \\ \text{between matrices)} \end{array}\tag{c}$$

2. Since the element at location (2, 2) is zero, interchange rows 2 and 3 to bring a nonzero element at that location (elementary row operation 1). Then dividing the new second row by 2 gives

$$\mathbf{B} \sim \begin{array}{ccc|c} x_1 & x_2 & x_3 & \mathbf{b} \\ \hline 1 & -1 & 1 & 0 \\ 0 & 1 & 0.5 & 2.5 \\ 0 & 0 & 1 & 1 \end{array}\tag{d}$$

3. Since the element at the location (3, 3) is one and all elements below the main diagonal are zero, the foregoing matrix puts the system of Eq. (a) into the form of Eq. (B.27), as

$$\begin{bmatrix} 1 & -1 & 1 \\ 0 & 1 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 2.5 \\ 1 \end{bmatrix} \quad (\text{e})$$

Performing the backward substitution, we obtain

$$x_3 = 1 \quad (\text{from third row})$$

$$x_2 = 2.5 - 0.5x_3 = 2 \quad (\text{from second row}) \quad (\text{f})$$

$$x_1 = 0 - x_3 + x_2 = 1 \quad (\text{from first row})$$

Therefore, the solution of Eq. (a) is

$$x_1 = 1, \quad x_2 = 2, \quad x_3 = 1 \quad (\text{g})$$

The Gaussian elimination method can easily be transcribed into a general-purpose computer program that can handle any given system of equations. However, certain modifications must be made to the procedure because numerical calculations on a machine with a finite number of digits introduce round-off errors. These errors can become significantly large if certain precautions are not taken. The modifications primarily involve a reordering of the rows or columns of the augmented matrix in such a way that possible round-off effects tend to be minimized. This reordering must be performed at each step of the elimination process so that the diagonal element of the pivot row is the absolute largest among the elements of the remaining matrix on the lower right side. This is known as the *total pivoting* procedure. When only the rows are interchanged to bring the absolute largest element from a column to the diagonal location, the procedure is known as *partial pivoting*. Note that many programs are available to solve a system of equations. Thus, before attempting to write a program for Gaussian elimination, computer center libraries must be searched for the existing ones.

EXAMPLE B.4 Determinant of a Matrix by Gaussian Elimination

The Gaussian elimination procedure can also be used to calculate the determinant of a matrix. We illustrate the procedure for the following 3×3 matrix:

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 0 \\ 1 & 2 & 1 \\ 0 & 3 & 4 \end{bmatrix} \quad (\text{a})$$

Solution. Using the Gaussian elimination procedure, we make the elements below the main diagonal zero, but this time the diagonal elements are not converted

to unity. Once the matrix is converted to that form, the determinant is obtained using Eq. (B.22).

$$\begin{bmatrix} 2 & 3 & 0 \\ 1 & 2 & 1 \\ 0 & 3 & 4 \end{bmatrix} \sim \begin{bmatrix} 2 & 3 & 0 \\ 0 & 0.5 & 1 \\ 0 & 3 & 4 \end{bmatrix} \quad \begin{array}{l} \text{(elimination in the first} \\ \text{column)} \end{array} \quad \text{(b)}$$

$$\sim \begin{bmatrix} 2 & 3 & 0 \\ 0 & 0.5 & 1 \\ 0 & 0 & -2 \end{bmatrix} \quad \begin{array}{l} \text{(elimination in the second} \\ \text{column)} \end{array} \quad \text{(c)}$$

The preceding system is in the canonical form, and $|\mathbf{A}|$ is given simply by multiplication of all the diagonal elements, i.e.,

$$|\mathbf{A}| = (2)(0.5)(-2) = -2. \quad \text{(d)}$$

B.3.4 Inverse of a Matrix: Gauss-Jordan Elimination

If the multiplication of two square matrices results in an identity matrix, they are called the inverse of each other. Let \mathbf{A} and \mathbf{B} be two square matrices of order n . Then \mathbf{B} is called the inverse of \mathbf{A} if

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}_{(n)} \quad \text{(B.30)}$$

The inverse of \mathbf{A} is usually denoted by \mathbf{A}^{-1} . We shall later describe methods to calculate the inverse of a matrix. Every square matrix may not have an inverse. A matrix having no inverse is called a *singular matrix*. If the coefficient matrix of an $n \times n$ system of equations has an inverse, then the system can be solved for the unknown variables. Consider the $n \times n$ system of equations $\mathbf{Ax} = \mathbf{b}$, where \mathbf{A} is the coefficient matrix and \mathbf{b} is the right-side vector. Pre-multiplying both sides of the equation by \mathbf{A}^{-1} , we get

$$\mathbf{A}^{-1}\mathbf{Ax} = \mathbf{A}^{-1}\mathbf{b}$$

Since $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$, the equation reduces to

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$$

Thus, if we know the inverse of matrix \mathbf{A} , then the preceding equation can be used to solve for the unknown vector \mathbf{x} .

There are a couple of ways to calculate the inverse of a nonsingular matrix. The first procedure is based on the use of the cofactors of \mathbf{A} and its determinant. If \mathbf{B} is the inverse of \mathbf{A} , then its elements are given as (called *inverse using cofactors*):

$$b_{ji} = \frac{\text{cofac}(a_{ij})}{|\mathbf{A}|}; \quad i = 1 \text{ to } n; j = 1 \text{ to } n \quad \text{(B.31)}$$

Note that indices on the left side of the equation are ji and on the right side they are ij . Thus, cofactors of the row of matrix \mathbf{A} generate the corresponding column of the inverse matrix \mathbf{B} .

The preceding procedure is reasonable for smaller matrices up to, say, 3×3 . For larger matrices, it becomes cumbersome and inefficient.

A clue to the second procedure for calculating the inverse is provided by Eq. (B.30). In that equation, elements of \mathbf{B} can be considered as unknowns for the system of equations $\mathbf{AB} = \mathbf{I}$. Thus, the system can be solved using the Gaussian elimination procedure to obtain the inverse of \mathbf{A} . We illustrate the procedure with an example.

EXAMPLE B.5 Inverse of a Matrix by Cofactors and Gauss-Jordan Reduction

Compute the inverse of the following 3×3 matrix:

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 0 \\ 1 & 2 & 0 \\ 0 & 3 & 1 \end{bmatrix} \quad (\text{a})$$

Solution.

Inverse by cofactors. Let \mathbf{B} be a 3×3 matrix that is the inverse of matrix \mathbf{A} . In order to use the *cofactors approach* given in Eq. (B.31), we first calculate the determinant of \mathbf{A} as $|\mathbf{A}| = -1$. Using Eq. (B.23), the cofactors of the first row of \mathbf{A} are

$$\text{cofac}(a_{11}) = (-1)^{1+1} \begin{vmatrix} 2 & 0 \\ 3 & 1 \end{vmatrix} = 2 \quad (\text{b})$$

$$\text{cofac}(a_{12}) = (-1)^{1+2} \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} = -1 \quad (\text{c})$$

$$\text{cofac}(a_{13}) = (-1)^{1+3} \begin{vmatrix} 1 & 2 \\ 0 & 3 \end{vmatrix} = 3 \quad (\text{d})$$

Similarly, the cofactors of the second and third rows are

$$-3, 1, -3; 0, 0, -1 \quad (\text{e})$$

Thus, Eq. (B.31) gives the inverse of \mathbf{A} as

$$\mathbf{B} = \begin{bmatrix} -2 & 3 & 0 \\ 1 & -1 & 0 \\ -3 & 3 & 1 \end{bmatrix} \quad (\text{f})$$

Inverse by Gaussian elimination. We shall first demonstrate the Gaussian elimination procedure before presenting the Gauss-Jordan procedure. Since \mathbf{B} is the inverse of \mathbf{A} , $\mathbf{AB} = \mathbf{I}$. Or writing this in the expanded form

$$\begin{bmatrix} 1 & 3 & 0 \\ 1 & 2 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (\text{g})$$

where b_{ij} 's are the elements of \mathbf{B} . The foregoing equation can be considered as a system of simultaneous equations having three different right-side vectors. We can solve for each unknown column on the left-side corresponding to each right-side vector by using the Gaussian elimination procedure. For example, considering the first column of \mathbf{B} only, we obtain

$$\begin{bmatrix} 1 & 3 & 0 \\ 1 & 2 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (\text{h})$$

Using the elimination procedure in the augmented matrix form, we obtain

$$\left[\begin{array}{ccc|c} 1 & 3 & 0 & 1 \\ 1 & 2 & 0 & 0 \\ 0 & 3 & 1 & 0 \end{array} \right] \sim \left[\begin{array}{ccc|c} 1 & 3 & 0 & 1 \\ 0 & -1 & 0 & -1 \\ 0 & 3 & 1 & 0 \end{array} \right] \quad \begin{array}{l} \text{(elimination in the} \\ \text{first column)} \end{array} \quad (\text{i})$$

$$\sim \left[\begin{array}{ccc|c} 1 & 3 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -3 \end{array} \right] \quad \begin{array}{l} \text{(elimination in the} \\ \text{second column)} \end{array} \quad (\text{j})$$

Using back substitution, we obtain the first column of \mathbf{B} as $b_{31} = -3$, $b_{21} = 1$, $b_{11} = -2$. Similarly, we find $b_{12} = 3$, $b_{22} = -1$, $b_{32} = 3$, $b_{13} = 0$, $b_{23} = 0$, and $b_{33} = 1$. Therefore, the inverse of \mathbf{A} is given as

$$\mathbf{B} = \begin{bmatrix} -2 & 3 & 0 \\ 1 & -1 & 0 \\ -3 & 3 & 1 \end{bmatrix} \quad (\text{k})$$

Inverse by Gauss-Jordan elimination. We can organize the procedure for calculating the inverse of a matrix slightly differently. The augmented matrix can be defined with all the three columns of the right side. The Gaussian elimination process can be carried out below as well as above the main diagonal. With this procedure, the left-hand 3×3 matrix is converted to an identity matrix; and the right-hand 3×3 matrix then contains the inverse of the matrix. When elimination is performed below as well as above the main diagonal, the procedure is called *Gauss-Jordan elimination*. The process proceeds as follows for calculating the inverse of \mathbf{A} :

$$\left[\begin{array}{ccc|ccc} 1 & 3 & 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 1 \end{array} \right] \quad \text{(augmented matrix)} \quad (\text{l})$$

$$\sim \left[\begin{array}{ccc|ccc} 1 & 3 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 1 \end{array} \right] \quad \begin{array}{l} \text{(elimination in the first column)} \\ \text{(m)} \end{array}$$

$$\sim \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -2 & 3 & 0 \\ 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -3 & 3 & 1 \end{array} \right] \quad \begin{array}{l} \text{(elimination in the second column)} \\ \text{(n)} \end{array}$$

There is no need to perform elimination on the third column since $\bar{a}_{13} = \bar{a}_{23} = 0$ and $\bar{a}_{33} = 1$. We observe from the above matrix that the last three columns give precisely the matrix \mathbf{B} , which is the inverse of \mathbf{A} .

The Gauss-Jordan procedure of computing the inverse of a 3×3 matrix can be generalized to any nonsingular $n \times n$ matrix. It can also be coded systematically into a general-purpose computer program to compute the inverse of a matrix.

B.4 Solution of m Linear Equations in n Unknowns

In the last section, the concept of determinants was used to determine the existence of a unique solution for any $n \times n$ system of equations. There are many instances in engineering applications where the number of equations is not equal to the number of variables, i.e., rectangular systems. In a system of m equations in n unknowns ($m \neq n$), the matrix of coefficients is not square. Therefore, no determinant can be associated with it. Thus, to treat such systems, a more general concept than the determinants is needed. We introduce such a concept in this section.

B.4.1 Rank of a Matrix

The general concept needed to develop the solution procedure for a general $m \times n$ system of equations is known as the *rank of the matrix*, defined as the order of the largest nonsingular square submatrix of the given matrix. Using the idea of a rank of a matrix, we can develop a general theory for the solution of a linear system of equations.

Let r be the rank of an $m \times n$ matrix \mathbf{A} . Then r satisfies the following conditions:

1. For $m < n$, $r \leq m < n$ (if $r = m$, the matrix is said to have full row rank).
2. For $n < m$, $r \leq n < m$ (if $r = n$, the matrix is said to have full column rank).
3. For $n = m$, $r \leq n$ (if $r = n$, the square matrix is called nonsingular).

In order to determine the rank of a matrix, we need to check the determinants of all the submatrices. This is a cumbersome and time-consuming process. However, it turns out that the Gauss-Jordan elimination process can be used to solve the linear system as well as determine the rank of the matrix. Using the Gauss-Jordan elimination procedure, we can transform any $m \times n$ matrix \mathbf{A} into the following equivalent form (for $m < n$):

$$\mathbf{A} \sim \left[\begin{array}{c|c} \mathbf{I}_{(r)} & \mathbf{0}_{(r \times n-r)} \\ \hline \mathbf{0}_{(m-r \times r)} & \mathbf{0}_{(m-r \times n-r)} \end{array} \right] \quad (\text{B.32})$$

where $\mathbf{I}_{(r)}$ is the $r \times r$ identity matrix. Then r is the rank of the matrix, where r satisfies one of the preceding three conditions. Note that the identity matrix $\mathbf{I}_{(r)}$ is unique for any given matrix.

EXAMPLE B.6 Rank Determination by Elementary Operations

Determine rank of the following matrix:

$$\mathbf{A} = \begin{bmatrix} 2 & 6 & 2 & 4 \\ -2 & -4 & 2 & 2 \\ 1 & 2 & -1 & -1 \end{bmatrix} \quad (\text{a})$$

Solution. The elementary operations lead to the following matrices:

$$\mathbf{A} \sim \begin{bmatrix} 1 & 3 & 1 & 2 \\ -2 & -4 & 2 & 2 \\ 1 & 2 & -1 & -1 \end{bmatrix} \quad \left(\text{multiply row 1 by } \frac{1}{2} \right) \quad (\text{b})$$

$$\sim \begin{bmatrix} 1 & 3 & 1 & 2 \\ 0 & 2 & 4 & 6 \\ 0 & -1 & -2 & -3 \end{bmatrix} \quad \begin{array}{l} \text{(add 2 times row 1 to row 2} \\ \text{and } -1 \text{ times row 1 to row 3)} \end{array} \quad (\text{c})$$

$$\sim \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 4 & 6 \\ 0 & -1 & -2 & -3 \end{bmatrix} \quad \begin{array}{l} \text{(add } -3 \text{ times column 1 to column 2;} \\ -1 \text{ times column 1 to column 3;} \\ -2 \text{ times column 1 to column 4)} \end{array} \quad (\text{d})$$

$$\sim \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \left(\text{multiply row 2 by } \frac{1}{2} \text{ and} \right. \\ \left. \text{add to row 3} \right) \quad (\text{e})$$

$$\sim \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{array}{l} \text{(add } -2 \text{ times column 2 to column 3;} \\ \text{and } -3 \text{ times column 2 to column 4)} \end{array} \quad (\text{f})$$

This matrix is in the form of Eq. (B.32). The rank of \mathbf{A} is 2, since a 2×2 identity matrix is obtained at the upper left corner.

B.4.2 General Solution of $m \times n$ Linear Equations

Let us now consider solving a system of m simultaneous equations in n unknowns. *The existence of a solution for such a system depends on the rank of the system's coefficient matrix and the augmented matrix.* Let the system be represented as

$$\mathbf{Ax} = \mathbf{b} \quad (\text{B.33})$$

where \mathbf{A} is an $m \times n$ matrix, \mathbf{b} is an m -vector, and \mathbf{x} is an n -vector of the unknowns. Note that m may be larger than n , i.e., there may be more equations than the number of unknowns. In that case the system is either *inconsistent* (has no solution) or some of the equations are redundant and may be deleted. The solution process described in the following provides answers to these questions.

Note that if an equation is multiplied by a constant, the solution of the system is unchanged. If c times one equation is added to another, the solution of the resulting system

is the same as for the original system. Also, if two columns of the coefficient matrix are interchanged (for example, columns i and j), the resulting set of equations is equivalent to the original system; however, the solution variables x_i and x_j are interchanged in the vector \mathbf{x} as follows:

$$\mathbf{x} = [x_1 \quad x_2 \quad \dots \quad x_{i-1} \quad \downarrow x_j \quad x_{i+1} \quad \dots \quad x_{j-1} \quad \downarrow x_i \quad x_{j+1} \quad \dots \quad x_n]^T \quad (\text{B.34})$$

This indicates that each column of the coefficient matrix has a variable associated with it as also noted earlier, e.g., x_i and x_j for the i th and the j th columns, respectively.

Using the elementary row-column operations, it is always possible to convert a system of m equations in n unknowns in Eq. (B.33) into an equivalent system of the form shown in the following Eq. (B.35). In the equation, a $\bar{}$ over each element indicates its new value, obtained by performing row-column operations on the augmented matrix of the original system. The value of subscript r in Eq. (B.35) is the rank of the coefficient matrix.

$$\left[\begin{array}{cccccccc|c} 1 & \bar{a}_{12} & \bar{a}_{13} & \bar{a}_{14} & \dots & \bar{a}_{1r} & \dots & \bar{a}_{1n} & x_1 \\ 0 & 1 & \bar{a}_{23} & \bar{a}_{24} & \dots & \bar{a}_{2r} & \dots & \bar{a}_{2n} & x_2 \\ 0 & 0 & 1 & \bar{a}_{34} & \dots & \bar{a}_{3r} & \dots & \bar{a}_{3n} & x_3 \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \dots & 1 & \dots & \bar{a}_{rn} & \cdot \\ 0 & 0 & 0 & 0 & \dots & 0 & \dots & 0 & \bar{b}_r \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \bar{b}_{r+1} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \dots & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & 0 & \dots & 0 & \dots & 0 & \bar{b}_m \end{array} \right] = \left[\begin{array}{c} \bar{b}_1 \\ \bar{b}_2 \\ \bar{b}_3 \\ \cdot \\ \cdot \\ \cdot \\ \bar{b}_r \\ \bar{b}_{r+1} \\ \cdot \\ \cdot \\ \cdot \\ \bar{b}_m \end{array} \right] \quad (\text{B.35})$$

Note that if $\bar{b}_{r+1} = \bar{b}_{r+2} = \dots = \bar{b}_m = 0$ in Eq. (B.35) then the last $(m - r)$ equations become

$$0x_1 + 0x_2 + \dots + 0x_n = 0$$

These rows can be eliminated from further consideration. However, if any of the last $(m - r)$ components of vector $\bar{\mathbf{b}}$ is not 0, then at least one of the last $(m - r)$ equations is *inconsistent* and the system has no solution. Note also that the rank of the coefficient matrix equals the rank of the augmented matrix if and only if $\bar{b}_i = 0$, $i = (r + 1)$ to m . Thus, a *system of m equations in n unknowns is consistent (i.e., possesses solutions) if and only if the rank of the coefficient matrix equals the rank of the augmented matrix.*

If elementary operations are performed below as above the main diagonal to eliminate off-diagonal elements, an equivalent system of the following form is obtained:

$$\left[\begin{array}{c|c} \mathbf{I}_{(r)} & \mathbf{Q}_{(r \times n-r)} \\ \hline \mathbf{0}_{(m-r \times r)} & \mathbf{0}_{(m-r \times n-r)} \end{array} \right] \left[\begin{array}{c} \mathbf{x}_{(r)} \\ \mathbf{x}_{(n-r)} \end{array} \right] = \left[\begin{array}{c} \mathbf{q}_{(r+1)} \\ \mathbf{p}_{(m-r \times 1)} \end{array} \right] \quad (\text{B.36})$$

Here $\mathbf{I}_{(r)}$ is an $r \times r$ identity matrix, and $\mathbf{x}_{(r)}$ and $\mathbf{x}_{(n-r)}$ are the r -component and $(n - r)$ -component subvectors of vector \mathbf{x} . Note that depending on the values of r , n , and m , the

equation can have several different forms. For example, if $r = n$, the matrices $\mathbf{Q}_{(r \times n-r)}$, $\mathbf{0}_{(m-r \times n-r)}$, and the vector $\mathbf{x}_{(n-r)}$ disappear; similarly, if $r = m$, matrices $\mathbf{0}_{(m-r \times r)}$, $\mathbf{0}_{(m-r \times n-r)}$, and the vector $\mathbf{p}_{(m-r \times 1)}$ disappear. The system of equations (B.33) is consistent only if vector $\mathbf{p} = \mathbf{0}$ in Eq. (B.36). *It must be remembered that for every interchange of columns necessary to produce Eq. (B.36), the corresponding components of \mathbf{x} must be interchanged.*

When the system is consistent, the first line of Eq. (B.36) gives

$$\mathbf{I}_{(r)}\mathbf{x}_{(r)} + \mathbf{Q}\mathbf{x}_{(n-r)} = \mathbf{q} \quad (\text{B.37})$$

or

$$\mathbf{x}_{(r)} = \mathbf{q} - \mathbf{Q}\mathbf{x}_{(n-r)} \quad (\text{B.38})$$

Equation (B.38) gives r components of \mathbf{x} in terms of the remaining $(n - r)$ components. If the system is consistent, Eq. (B.38) represents the *general solution* of the system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$. The last $(n - r)$ components of \mathbf{x} can be assigned arbitrary values; any assignment to x_{r+1}, \dots, x_n yields a solution. Thus, the system of equations has infinitely many solutions. If $r = n$, the solution is unique. Equation (B.36) is known as the *canonical representation* for the system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$. This form of equations is very useful in solving linear programming problems in Chapter 4.

The following examples illustrate the Gauss-Jordan elimination procedure.

EXAMPLE B.7 General Solution by Gauss-Jordan Reduction

Find a general solution for the set of equations

$$\begin{aligned} x_1 + x_2 + x_3 + 5x_4 &= 6 \\ x_1 + x_2 - 2x_3 - x_4 &= 0 \\ x_1 + x_2 - x_3 + x_4 &= 2 \end{aligned} \quad (\text{a})$$

Solution. The augmented matrix for the set of equations is given as

$$\mathbf{A} \sim \begin{array}{cccc|c} x_1 & x_2 & x_3 & x_4 & \mathbf{b} \\ 1 & 1 & 1 & 5 & 6 \\ 1 & 1 & -2 & -1 & 0 \\ 1 & 1 & -1 & 1 & 2 \end{array} \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad (\text{b})$$

The following elimination steps are used to transform the system to a canonical form:

1. Subtracting row 1 from rows 2 and 3, we convert elements below the main diagonal in the first column (a_{21} and a_{31}) to zero, i.e., we eliminate x_1 from equations 2 and 3, and obtain

$$\mathbf{A} \sim \left[\begin{array}{cccc|c} x_1 & x_2 & x_3 & x_4 & \mathbf{b} \\ 1 & 1 & 1 & 5 & 6 \\ 1 & 1 & -3 & -6 & -6 \\ 1 & 1 & -2 & -4 & -4 \end{array} \right] \quad (\text{c})$$

2. Now, since a_{22} is zero we cannot proceed any further with the elimination process. We must interchange rows and/or columns to bring a nonzero element at location a_{22} . We can interchange either column 3 or column 4 with column 2 to bring a nonzero element in the position a_{22} . (Note: The last column can never be interchanged with any other column; it is the right side of the system $\mathbf{Ax} = \mathbf{b}$, so it does not correspond to a variable). Interchanging column 2 with column 3 (elementary column operation 1), we obtain

$$\mathbf{A} \sim \left[\begin{array}{cccc|c} x_1 & x_2 & x_3 & x_4 & \mathbf{b} \\ 1 & 1 & 1 & 5 & 6 \\ 0 & -3 & 0 & -6 & -6 \\ 0 & -2 & 0 & -4 & -4 \end{array} \right] \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_3 \\ x_2 \\ x_4 \end{bmatrix} \quad (\text{d})$$

Note that the positions of the variables x_2 and x_3 are also interchanged in the vector \mathbf{x} .

3. Now, dividing row 2 by -3 , multiplying it by 2 and adding to row 3 gives

$$\mathbf{A} \sim \left[\begin{array}{cccc|c} x_1 & x_3 & x_2 & x_4 & \mathbf{b} \\ 1 & 1 & 1 & 5 & 6 \\ 0 & 1 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right] \quad (\text{e})$$

Thus elements below the main diagonal in Eq. (e) are zero and the Gaussian elimination process is complete.

4. To put the equations in the canonical form of Eq. (B.36), we need to perform elimination above the main diagonal also (Gauss-Jordan elimination). Subtracting row 2 from row 1, we obtain

$$\mathbf{A} \sim \left[\begin{array}{cccc|c} x_1 & x_3 & x_2 & x_4 & \mathbf{b} \\ 1 & 0 & 1 & 3 & 4 \\ 0 & 1 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{array} \right] \quad (\text{f})$$

5. Using the matrix of Eq. (f), the given system of equations is transformed into a canonical form of Eq. (B.36) as follows:

$$\left[\begin{array}{ccc|c} 1 & 0 & 1 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{array} \right] \begin{bmatrix} x_1 \\ x_3 \\ x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 0 \end{bmatrix} \quad (\text{g})$$

or

$$\left[\begin{array}{c|c} \mathbf{I}_{(2)} & \mathbf{Q}_{(2 \times 2)} \\ \hline \mathbf{0}_{(1 \times 2)} & \mathbf{0}_{(1 \times 2)} \end{array} \right] \begin{bmatrix} x_1 \\ x_3 \\ x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} \mathbf{q}_{(2 \times 1)} \\ \mathbf{p}_{(1 \times 1)} \end{bmatrix} \quad (\text{h})$$

where

$$\mathbf{Q} = \begin{bmatrix} 1 & 3 \\ 0 & 2 \end{bmatrix}; \quad \mathbf{q} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}; \quad \mathbf{p} = \mathbf{0} \quad (\text{i})$$

$$\mathbf{x}_{(r)} = (x_1, x_3), \quad \mathbf{x}_{(n-r)} = (x_2, x_4)$$

6. Since $\mathbf{p} = \mathbf{0}$, the given system of equations is consistent (i.e., it has solutions). Its general solution, in the form of Eq. (B.38) is

$$\begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix} - \begin{bmatrix} 1 & 3 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} \quad (\text{j})$$

Or, in the expanded notation, the general solution is

$$\begin{aligned} x_1 &= 4 - x_2 - 3x_4 \\ x_3 &= 2 - 2x_4 \end{aligned} \quad (\text{k})$$

7. It can be seen that the general solution of Eq. (k) gives x_1 and x_3 in terms of x_2 and x_4 ; i.e., x_2 and x_4 are *independent variables* and x_1 and x_3 are *dependent* on them. The system has infinite solutions because any specification for x_2 and x_4 gives a solution.

Basic Solutions In the preceding general solution, we see that x_2 and x_4 can be given arbitrary values, and the corresponding x_1 and x_3 calculated from the Eq. (k). Thus the system has an infinite number of solutions. A particular solution of much interest in linear programming (LP) is obtained by setting $\mathbf{x}_{(n-r)} = \mathbf{0}$ in the general solution of Eq. (B.38). Such a solution is called the *basic solution* of the system of equations $\mathbf{Ax} = \mathbf{b}$. For the present example, a basic solution is $x_1 = 4$, $x_2 = 0$, $x_3 = 2$, and $x_4 = 0$, which is obtained from Eq. (k) by setting $x_2 = x_4 = 0$.

Note that although Eq. (k) give an infinite number of solutions for the system of equations, the number of basic solutions is finite. For example, another basic solution can be obtained by setting $x_2 = x_3 = 0$ and solving for x_1 and x_4 . It can be verified that this basic solution is $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, and $x_4 = 1$. The fact that the number of basic solutions is finite is very important for the linear programming problems discussed in Chapter 6. The reason is that the *optimum solution for an LP problem is one of the basic solutions*.

Since the last equation essentially gives $0 = 0$, the given system of equations is consistent (i.e., it has solutions). Also, since the rank of the coefficient matrix is 3, which is less than the number of equations, there are an infinite number of solutions for the linear system. From the preceding equation the general solution is given as

$$\begin{aligned}x_1 &= 2 - x_3 \\x_2 &= 0 + x_3 \\x_4 &= 1\end{aligned}\tag{c}$$

A basic solution is obtained by setting x_3 to zero as $x_1 = 2$, $x_2 = 0$, $x_3 = 0$, $x_4 = 1$.

To summarize the results of this section, we note that

1. The $m \times n$ system of equations (B.33) is *consistent* if the rank of the coefficient matrix is the same as the rank of the augmented matrix. A consistent system implies that it has a solution.
2. If the number of equations is less than the number of variables ($m < n$) and the system is consistent, having rank less than or equal to m ($r \leq m$), then it has infinitely many solutions.
3. If $m = n = r$, then the system (B.33) has a unique solution.

B.5 Concepts Related to a Set of Vectors

In several applications, we come across a set of vectors. It is useful to discuss some concepts related to these sets, such as the linear independence of vectors and vector spaces. In this section, we briefly discuss these concepts and describe a procedure for checking the linear independence of a set of vectors.

B.5.1 Linear Independence of a Set of Vectors

Consider a set of k vectors each of dimension n :

$$A = \{\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(k)}\}$$

where a superscript (i) represents the i th vector. A *linear combination* of vectors in the set A is another vector obtained by scaling each vector in A and adding all the resulting vectors. That is, if \mathbf{b} is a linear combination of a vector in A , then it is defined as

$$\mathbf{b} = x_1 \mathbf{a}^{(1)} + x_2 \mathbf{a}^{(2)} + \dots + x_k \mathbf{a}^{(k)} = \sum_{i=1}^k x_i \mathbf{a}^{(i)}\tag{B.39a}$$

where x_1, x_2, \dots, x_k are some scalars. The preceding equation can be written compactly in matrix form as

$$\mathbf{b} = \mathbf{A}\mathbf{x}\tag{B.39b}$$

where \mathbf{x} is a k -component vector and \mathbf{A} is an $n \times k$ matrix with vectors $\mathbf{a}^{(i)}$ as its columns.

To determine if the set of vectors is *linearly independent* or *dependent*, we set the linear combination of Eq. (B.39) to zero as

$$x_1 \mathbf{a}^{(1)} + x_2 \mathbf{a}^{(2)} + \dots + x_k \mathbf{a}^{(k)} = \mathbf{0}; \quad \text{or} \quad \mathbf{A}\mathbf{x} = \mathbf{0} \quad (\text{B.40})$$

This gives a homogeneous system of equations with x_i 's as unknowns. There are n equations in k unknowns. Note that $\mathbf{x} = \mathbf{0}$ satisfies Eq. (B.40). *If $\mathbf{x} = \mathbf{0}$ is the only solution, then the set of vectors is linearly independent. In this case rank r of the matrix \mathbf{A} must be equal to k (the number of vectors in the set). If there exists a set of scalars x_i not all zero and satisfying Eq. (B.40), then the vectors $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(k)}$ are said to be linearly dependent. In this case rank r of \mathbf{A} is less than k .*

If a set of vectors is linearly dependent, then one or more vectors are parallel to each other, or there is at least one vector that can be expressed as a linear combination of the rest. That is, at least one of the scalars x_1, x_2, \dots, x_k must be nonzero. If we assume x_j to be nonzero, then Eq. (B.40) can be written as follows:

$$\begin{aligned} -x_j \mathbf{a}^{(j)} &= x_1 \mathbf{a}^{(1)} + x_2 \mathbf{a}^{(2)} + \dots + x_{j-1} \mathbf{a}^{(j-1)} + x_{j+1} \mathbf{a}^{(j+1)} \\ &+ \dots + x_k \mathbf{a}^{(k)} = \sum_{i=1}^k x_i \mathbf{a}^{(i)}; \quad i \neq j \end{aligned}$$

Or, since $x_j \neq 0$, we can divide both sides by it to obtain

$$\mathbf{a}^{(j)} = -\sum_{i=1}^k (x_i/x_j) \mathbf{a}^{(i)}; \quad i \neq j \quad (\text{B.41})$$

In Eq. (B.41) we have expressed $\mathbf{a}^{(j)}$ as a *linear combination* of $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(j-1)}, \mathbf{a}^{(j+1)}, \dots, \mathbf{a}^{(k)}$. In general, we see that if a set of vectors is linearly dependent, then at least one of them can be expressed as a linear combination of the rest.

EXAMPLE B.9 Check for Linear Independence of Vectors

Check linear independence of the following set of vectors:

$$(i) \quad \mathbf{a}^{(1)} = \begin{bmatrix} 2 \\ 5 \\ 2 \\ -1 \end{bmatrix}, \quad \mathbf{a}^{(2)} = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{a}^{(3)} = \begin{bmatrix} 8 \\ 9 \\ 4 \\ -1 \end{bmatrix}$$

$$(ii) \quad \mathbf{a}^{(1)} = \begin{bmatrix} 2 \\ 6 \\ 2 \\ -2 \end{bmatrix}, \quad \mathbf{a}^{(2)} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 0 \end{bmatrix}, \quad \mathbf{a}^{(3)} = \begin{bmatrix} 6 \\ 9 \\ 4 \\ 1 \end{bmatrix}$$

Solution. To check for linear independence, we form the linear combination of Eq. (B.39) and set it to zero as in Eq. (B.40). The resulting homogeneous system of

equations is solved for the scalars x_i . If all the scalars are zero, then the given set of vectors is linearly independent; otherwise it is dependent.

The vectors in set (i) are linearly dependent, since $x_1 = 1$, $x_2 = 2$, and $x_3 = -1$ give the linear combination of Eq. (B.40) a zero value, i.e.,

$$\mathbf{a}^{(1)} + 2\mathbf{a}^{(2)} - \mathbf{a}^{(3)} = \mathbf{0} \quad (\text{a})$$

It may also be checked that the rank of the following matrix whose columns are the given vectors is only 2; so the set of vectors is linearly dependent:

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 8 \\ 5 & 2 & 9 \\ 2 & 1 & 4 \\ -1 & 0 & -1 \end{bmatrix} \quad (\text{b})$$

For set (ii), let us form a linear combination of the given vectors and set it to zero:

$$x_1\mathbf{a}^{(1)} + x_2\mathbf{a}^{(2)} + x_3\mathbf{a}^{(3)} = \mathbf{0} \quad (\text{c})$$

This is a vector equation that gives the following system when written in the expanded form:

$$2x_1 + 4x_2 + 6x_3 = 0 \quad (\text{d})$$

$$6x_1 + 3x_2 + 9x_3 = 0 \quad (\text{e})$$

$$2x_1 + 2x_2 + 4x_3 = 0 \quad (\text{f})$$

$$-2x_1 \quad + \quad x_3 = 0 \quad (\text{g})$$

We solve the preceding system of equations by the elimination process.

From Eq. (g), we find $x_3 = 2x_1$. Equations (d) to (f) then become

$$14x_1 + 4x_2 = 0 \quad (\text{h})$$

$$24x_1 + 3x_2 = 0 \quad (\text{i})$$

$$10x_1 + 2x_2 = 0 \quad (\text{j})$$

From Eq. (j), we find $x_2 = -5x_1$. Substituting this result into Eqs. (h) and (i) gives

$$14x_1 + 4(-5x_1) = -6x_1 = 0 \quad (\text{k})$$

$$24x_1 + 3(-5x_1) = 9x_1 = 0 \quad (\text{l})$$

Equations (k) and (l) imply $x_1 = 0$; therefore, $x_2 = -5x_1 = 0$, $x_3 = 2x_1 = 0$. Thus, the only solution to Eq. (c) is the trivial solution $x_1 = x_2 = x_3 = 0$. The vectors $\mathbf{a}^{(1)}$, $\mathbf{a}^{(2)}$, and $\mathbf{a}^{(3)}$ are therefore linearly independent.

Equation (B.40) may be considered as a set of n simultaneous equations in k unknowns. To see this, define the k vectors as

$$\mathbf{a}^{(1)} = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \vdots \\ a_{n1} \end{bmatrix}, \quad \mathbf{a}^{(2)} = \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \\ \vdots \\ a_{n2} \end{bmatrix}, \quad \dots, \quad \mathbf{a}^{(k)} = \begin{bmatrix} a_{1k} \\ a_{2k} \\ a_{3k} \\ \vdots \\ a_{nk} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}$$

Also, let $\mathbf{A}_{(n \times k)} = [\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(k)}]$, i.e., \mathbf{A} is a matrix whose i th column is the i th vector $\mathbf{a}^{(i)}$. Then Eq. (B.40) can be written as

$$\mathbf{Ax} = \mathbf{0} \tag{B.42}$$

The results of Section B.4 show that there is a unique solution to Eq. (B.42) if and only if the rank r of \mathbf{A} is equal to k ($r = k < n$), the number of columns of \mathbf{A} . In that case, the unique solution is $\mathbf{x} = \mathbf{0}$. Therefore, *the vectors $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(k)}$ are linearly independent if and only if the rank of the matrix \mathbf{A} is k* (the number of vectors in the set).

Note that if $k > n$, then the rank of \mathbf{A} cannot exceed n . Therefore, *$\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(k)}$ will always be linearly dependent if $k > n$* . Thus, the maximum number of linearly dependent n -component vectors is n . Any set of $(n + 1)$ vectors is always linearly dependent.

Given any set of n linearly independent (n -component) vectors, $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(n)}$, any other (n -component) vector \mathbf{b} can be expressed as a unique linear combination of these vectors. The problem is to choose a set of scalars x_1, x_2, \dots, x_n such that

$$x_1 \mathbf{a}^{(1)} + x_2 \mathbf{a}^{(2)} + \dots + x_n \mathbf{a}^{(n)} = \mathbf{b}; \quad \text{or} \quad \mathbf{Ax} = \mathbf{b} \tag{B.43}$$

We wish to show that a solution exists for Eq. (B.43) and it is unique. Note that $\mathbf{a}^{(1)}, \mathbf{a}^{(2)}, \dots, \mathbf{a}^{(n)}$ are linearly independent. Therefore, the rank of the coefficient matrix \mathbf{A} is n , and the rank of the augmented matrix $[\mathbf{A}, \mathbf{b}]$ is also n . It cannot be $(n + 1)$ because the matrix has only n rows. Thus, Eq. (B.43) always possesses a solution for any given \mathbf{b} . Moreover, \mathbf{A} is nonsingular, hence the solution is unique.

In summary, we state the following points for a k set of vectors each having n components:

1. If $k > n$, the set of vectors is always linearly dependent, e.g., three vectors each having two components. That is, the number of linearly independent vectors is always less than or equal to n , e.g., for two-component vectors, there are at the most two linearly independent vectors.
2. If there are n linearly independent vectors each of dimension n , then any other n -component vector can be expressed as a unique linear combination of them, e.g., given two linearly independent vectors $\mathbf{a}^{(1)} = (1, 0)$ and $\mathbf{a}^{(2)} = (0, 1)$ of dimension two, any other vector such as $\mathbf{b} = (b_1, b_2)$ can be expressed as a unique linear combination of $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$.
3. Linear independence of the given set of vectors can be determined in two ways:
 - (i) Form the matrix \mathbf{A} of dimension $n \times k$ whose columns are the given vectors. Then, if rank r is equal to k ($r = k$), the given set is linearly independent; otherwise it is dependent.

- (ii) Set the linear combination of the given vectors to zero as $\mathbf{Ax} = \mathbf{0}$. If $\mathbf{x} = \mathbf{0}$ is the only solution for the resulting system, then the set is independent; otherwise it is dependent.

B.5.2 Vector Spaces

Before defining the concept of a vector space, let us define closure under addition and scalar multiplication:

Definition: Closure under addition. A set of vectors is said to be closed under addition if the sum of any two vectors in the set is also in the set.

Definition: Closure under scalar multiplication. A set of vectors is said to be closed under scalar multiplication if the product of any vector in the set by a scalar gives a vector in the set.

Definition: Vector space. A nonempty set S of elements (vectors) $\mathbf{x}, \mathbf{y}, \mathbf{z}, \dots$ is called a vector space if the two algebraic operations (vector addition and multiplication by a real scalar) on them satisfy the following properties:

1. Closure under addition: if $\mathbf{x} \in S$ and $\mathbf{y} \in S$ then $\mathbf{x} + \mathbf{y} \in S$.
2. Commutative in addition: $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$.
3. Associative in addition: $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$.
4. Identity in addition: there exists a zero vector $\mathbf{0}$ in the set S such that $\mathbf{x} + \mathbf{0} = \mathbf{x}$ for all \mathbf{x} .
5. Inverse in addition: there exists a $-\mathbf{x}$ in the set S such that $\mathbf{x} + (-\mathbf{x}) = \mathbf{0}$ for all \mathbf{x} .
6. Closure under scalar multiplication: for real scalars α, β, \dots , if $\mathbf{x} \in S$ then $\alpha\mathbf{x} \in S$.
7. Distributive: $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$.
8. Distributive: $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{x} + \alpha\mathbf{y}$.
9. Associative in scalar multiplication: $(\alpha\beta)\mathbf{x} = \alpha(\beta\mathbf{x})$.
10. Identity in scalar multiplication: $1\mathbf{x} = \mathbf{x}$.

In the preceding section, it was noted that the maximum number of linearly independent vectors in the set of all n -component vectors is n . Thus, for every subset of this set, there exists some maximum number of linearly independent vectors. In particular, every vector space has a maximum number of linearly independent vectors. This number is called the *dimension of the vector space*. If a vector space has dimension k , then any set of k linearly independent vectors in the vector space is called a *basis* for the vector space. Any other vector in the vector space can be expressed as a unique linear combination of the given set of basis vectors.

EXAMPLE B.10 Check for Vector Space

Check if the set $S = \{(x_1, x_2, x_3) \mid x_1 = 0\}$ is a vector space.

Solution. To see this consider any two vectors in S as

$$\mathbf{x} = \begin{bmatrix} 0 \\ a \\ b \end{bmatrix} \quad \text{and} \quad \mathbf{y} = \begin{bmatrix} 0 \\ c \\ d \end{bmatrix} \quad (\text{a})$$

where scalars $a, b, c,$ and d are completely arbitrary. Then,

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} 0 \\ a+c \\ b+d \end{bmatrix} \quad (\text{b})$$

Therefore, $\mathbf{x} + \mathbf{y}$ is in the set S . Also, for any scalar α ,

$$\alpha \mathbf{x} = \begin{bmatrix} 0 \\ \alpha a \\ \alpha b \end{bmatrix} \quad (\text{c})$$

Therefore, $\alpha \mathbf{x}$ is in the set S . Thus, S is closed under addition and scalar multiplication. All other properties for the definition of a vector space can be proved easily. To show the property (2), we have

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} 0 \\ a+c \\ b+d \end{bmatrix} = \begin{bmatrix} 0 \\ c+a \\ d+b \end{bmatrix} = \begin{bmatrix} 0 \\ c \\ d \end{bmatrix} + \begin{bmatrix} 0 \\ a \\ b \end{bmatrix} = \mathbf{y} + \mathbf{x} \quad (\text{d})$$

“Associative in addition” is shown as

$$\begin{aligned} (\mathbf{x} + \mathbf{y}) + \mathbf{z} &= \begin{bmatrix} 0 \\ a+c \\ b+d \end{bmatrix} + \begin{bmatrix} 0 \\ e \\ f \end{bmatrix} = \begin{bmatrix} 0 \\ a+c+e \\ b+d+f \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ a \\ b \end{bmatrix} + \begin{bmatrix} 0 \\ c+e \\ d+f \end{bmatrix} = \mathbf{x} + (\mathbf{y} + \mathbf{z}). \end{aligned} \quad (\text{e})$$

For identity in addition we have a zero vector in the set S as

$$\mathbf{0} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad (\text{f})$$

such that

$$\mathbf{x} + \mathbf{0} = \begin{bmatrix} 0 \\ a+0 \\ b+0 \end{bmatrix} = \begin{bmatrix} 0 \\ a \\ b \end{bmatrix} = \mathbf{x} \quad (\text{g})$$

Inverse in addition exists if we define $-\mathbf{x}$ as

$$-\mathbf{x} = -\begin{bmatrix} 0 \\ a \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ -a \\ -b \end{bmatrix} \quad (\text{h})$$

such that

$$\mathbf{x} + (-\mathbf{x}) = \begin{bmatrix} 0 \\ a+(-a) \\ b+(-b) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \mathbf{0} \quad (\text{i})$$

In a similar way, properties (7) to (10) can easily be shown.

Therefore, the set S is a vector space. Note that the set $V = \{(x_1, x_2, x_3) \mid x_1 = 1\}$ is not a vector space.

Let us now determine the dimension of S . Note that if \mathbf{A} is a matrix whose columns are vectors in S , then \mathbf{A} has three rows, and the first row contains only zeros. Thus, the rank of \mathbf{A} must be less than or equal to 2, and the dimension of S is either 1 or 2. To show that it is in fact 2, we need only find two linearly independent vectors. The following are three such sets of two linearly independent vectors from the set S :

$$\text{(i) } \mathbf{a}^{(1)} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{a}^{(2)} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (\text{j})$$

$$\text{(ii) } \mathbf{a}^{(3)} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}, \quad \mathbf{a}^{(4)} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (\text{k})$$

$$\text{(iii) } \mathbf{a}^{(5)} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{a}^{(6)} = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \quad (\text{l})$$

Each of these three sets is a basis for S . Any vector in S can be expressed as a linear combination of each of these sets. If $\mathbf{x} = (0, c, d)$ is any element of S , then

$$\text{(i) } \mathbf{x} = c\mathbf{a}^{(1)} + d\mathbf{a}^{(2)}$$

$$\begin{bmatrix} 0 \\ c \\ d \end{bmatrix} = c \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + d \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (\text{m})$$

$$\text{(ii) } \mathbf{x} = \frac{c}{2}\mathbf{a}^{(3)} + \left(c - \frac{d}{2}\right)\mathbf{a}^{(4)}$$

$$\begin{bmatrix} 0 \\ c \\ d \end{bmatrix} = \frac{c}{2} \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} + \left(c - \frac{d}{2}\right) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (\text{n})$$

$$\text{(iii) } \mathbf{x} = \left(\frac{c+d}{2}\right)\mathbf{a}^{(5)} + \left(\frac{c-d}{2}\right)\mathbf{a}^{(6)}$$

$$\begin{bmatrix} 0 \\ c \\ d \end{bmatrix} = \frac{c+d}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + \frac{c-d}{2} \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} \quad (\text{o})$$

B.6 Eigenvalues and Eigenvectors

Given an $n \times n$ matrix \mathbf{A} , any nonzero vector \mathbf{x} satisfying

$$\mathbf{Ax} = \lambda\mathbf{x} \quad (\text{B.44})$$

where λ is a scale factor, is called an *eigenvector* (proper or characteristic vector). The scalar λ is called the *eigenvalue* (proper or characteristic value). Since $\mathbf{x} \neq \mathbf{0}$, from Eq. (B.44) we see that λ is given as roots of the characteristic equation

$$|\mathbf{A} - \lambda\mathbf{I}| = 0 \quad (\text{B.45})$$

Equation (B.45) gives an n th degree polynomial in λ . Roots of this polynomial are the required eigenvalues. After eigenvalues have been determined, eigenvectors can be determined from Eq. (B.44).

The coefficient matrix \mathbf{A} may be symmetric or asymmetric. For many applications, \mathbf{A} is a symmetric matrix, so we consider this case in the text. Some properties of eigenvalues and eigenvectors are:

1. Eigenvalues and eigenvectors of a real symmetric matrix are real. They may be complex for real nonsymmetric matrices.
2. Eigenvectors corresponding to distinct eigenvalues of real symmetric matrices are orthogonal to each other (that is, their dot product vanishes).

EXAMPLE B.11 Calculation of Eigenvalues and Eigenvectors

Find eigenvalues and eigenvectors of the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad (\text{a})$$

Solution. The eigenvalue problem is defined as

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (\text{b})$$

The characteristic polynomial is given by $|\mathbf{A} - \lambda\mathbf{I}| = 0$,

$$\begin{vmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{vmatrix} = 0 \quad (\text{c})$$

or,

$$\lambda^2 - 4\lambda + 3 = 0 \quad (\text{d})$$

The roots of this polynomial are

$$\lambda_1 = 3, \quad \lambda_2 = 1 \quad (\text{e})$$

Therefore, the eigenvalues are 3 and 1.

The eigenvectors are determined from Eq. (B.44). For $\lambda_1 = 3$, Eq. (B.44) is

$$\begin{bmatrix} (2-3) & 1 \\ 1 & (2-3) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{f})$$

or, $x_1 = x_2$. Therefore, a solution of the above equation is (1, 1). After normalization (dividing by its length), the first eigenvector becomes

$$\mathbf{x}^{(1)} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (\text{g})$$

For $\lambda_2 = 1$, Eq. (B.44) is

$$\begin{bmatrix} (2-1) & 1 \\ 1 & (2-1) \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{h})$$

or $x_1 = -x_2$. Therefore, a solution of the above equation is (1, -1). After normalization, the second eigenvector is

$$\mathbf{x}^{(2)} = \left(\frac{1}{\sqrt{2}} \right) \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad (\text{i})$$

It may be verified that $\mathbf{x}^{(1)} \cdot \mathbf{x}^{(2)}$ is zero, i.e., $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are orthogonal to each other.

B.7* Norm and Condition Number of a Matrix

B.7.1 Norm of Vectors and Matrices

Every n -dimensional vector \mathbf{x} has a scalar-valued function associated with it, denoted as $\|\mathbf{x}\|$. It is called a norm of \mathbf{x} if it satisfies the following three conditions:

1. $\|\mathbf{x}\| > 0$ for $\mathbf{x} \neq \mathbf{0}$, and $\|\mathbf{x}\| = 0$ only when $\mathbf{x} = \mathbf{0}$.
2. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ (triangle inequality).
3. $\|\mathbf{a}\mathbf{x}\| = |\mathbf{a}| \|\mathbf{x}\|$ where a is a scalar.

The ordinary length of a vector for $n \leq 3$ satisfies the foregoing three conditions. The concept of norm is therefore a generalization of the ordinary length of a vector in one-, two-, or three-dimensional Euclidean space. For example, it can be verified that the Euclidean distance in the n -dimensional space

$$\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}} = \sqrt{\mathbf{x} \cdot \mathbf{x}} \quad (\text{B.46})$$

satisfies the three norm conditions and hence is a norm.

Every $n \times n$ matrix \mathbf{A} has a *scalar function* associated with it called its norm. It is denoted as $\|\mathbf{A}\|$ and is calculated as

$$\|\mathbf{A}\| = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \quad (\text{B.47})$$

Note that since $\mathbf{A}\mathbf{x}$ is a vector, Eq. (B.47) says that the norm of \mathbf{A} is determined by the vector \mathbf{x} that maximizes the ratio $\|\mathbf{A}\mathbf{x}\|/\|\mathbf{x}\|$. The three conditions of the norm can be verified easily for Eq. (B.47), as follows:

1. $\|\mathbf{A}\| > 0$ unless it is a null matrix in which case it is zero.
2. $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$.
3. $\|a\mathbf{A}\| = |a| \|\mathbf{A}\|$ where a is a scalar.

Other vector norms can also be defined. For example, the summation norm and the max-norm (called the “ ∞ -norm”) are defined as

$$\|\mathbf{x}\| = \sum_{i=1}^n |x_i|, \quad \text{or} \quad \|\mathbf{x}\| = \max_{1 \leq i \leq n} |x_i| \quad (\text{B.48})$$

They also satisfy the three conditions of the norm of the vector \mathbf{x} .

If λ_1^2 is the largest eigenvalue of $\mathbf{A}^T\mathbf{A}$, then it can be shown using Eq. (B.47) that the norm of \mathbf{A} is also defined as

$$\|\mathbf{A}\| = \lambda_1 > 0$$

Similarly, if λ_n^2 is the smallest eigenvalue of $\mathbf{A}^T\mathbf{A}$, then the norm of \mathbf{A}^{-1} is defined as

$$\|\mathbf{A}^{-1}\| = \lambda_n > 0$$

B.7.2 Condition Number of a Matrix

The condition number is another scalar associated with an $n \times n$ matrix. The idea of a condition number is useful while solving a linear system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$. Often there is uncertainty in elements of the coefficient matrix \mathbf{A} or the right side vector \mathbf{b} . The question then is, how does the solution vector \mathbf{x} change for small perturbations in \mathbf{A} and \mathbf{b} ? The answer to this question is contained in the condition number of the matrix \mathbf{A} .

It can be shown that the condition number of an $n \times n$ matrix \mathbf{A} , denoted as $\text{cond}(\mathbf{A})$, is given as

$$\text{cond}(\mathbf{A}) = \lambda_1 / \lambda_n \geq 0$$

where λ_1^2 and λ_n^2 are the largest and the smallest eigenvalues of $\mathbf{A}^T\mathbf{A}$. It turns out that a larger condition number indicates that the solution \mathbf{x} is very sensitive to variations in the elements of \mathbf{A} and \mathbf{b} . That is, small changes in \mathbf{A} and \mathbf{b} give large changes in \mathbf{x} .

A very large condition number for the matrix \mathbf{A} indicates it to be nearly singular. The corresponding system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ is called ill-conditioned.

Exercises for Appendix B

Evaluate the following determinants.

$$\text{B.1 } \begin{vmatrix} 2 & 1 & 3 \\ 1 & 2 & 1 \\ 3 & 1 & 5 \end{vmatrix} \quad \text{B.2 } \begin{vmatrix} 0 & 2 & 3 & 2 \\ 0 & 4 & 5 & 4 \\ 1 & -2 & -2 & 1 \\ 3 & -1 & 2 & 1 \end{vmatrix}$$

$$\text{B.3 } \begin{vmatrix} 0 & 0 & 0 & -2 \\ 0 & 0 & 5 & 3 \\ 0 & 1 & -1 & 1 \\ 2 & 3 & -3 & 2 \end{vmatrix}$$

For the following determinants, calculate values of the scalar λ for which determinants vanish.

$$\text{B.4 } \begin{vmatrix} 2-\lambda & 1 & 0 \\ 1 & 3-\lambda & 0 \\ 0 & 3 & 2-\lambda \end{vmatrix} \quad \text{B.5 } \begin{vmatrix} 2-\lambda & 2 & 0 \\ 1 & 2-\lambda & 0 \\ 0 & 0 & 2-\lambda \end{vmatrix}$$

Determine the rank of the following matrices.

$$\text{B.6 } \begin{bmatrix} 3 & 0 & 1 & 3 \\ 2 & 0 & 3 & 2 \\ 0 & 2 & -8 & 1 \\ -2 & -1 & 2 & -1 \end{bmatrix} \quad \text{B.7 } \begin{bmatrix} 1 & 2 & 2 & 2 & 4 \\ 1 & 6 & 3 & 0 & 3 \\ 2 & 2 & 3 & 3 & 2 \\ 1 & 3 & 2 & 5 & 1 \end{bmatrix}$$

$$\text{B.8 } \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 0 & 0 & 1 \\ 3 & 2 & 3 & 0 \\ 2 & 3 & 1 & 4 \\ 2 & 0 & 6 & 0 \\ 1 & 2 & 1 & 4 \end{bmatrix}$$

Obtain the solution of the following equations using the Gaussian elimination procedure.

$$\text{B.9 } \begin{aligned} 2x_1 + 2x_2 + x_3 &= 5 \\ x_1 - 2x_2 + 2x_3 &= 1 \\ x_2 + 2x_3 &= 3 \end{aligned}$$

$$\text{B.10 } \begin{aligned} x_2 - x_3 &= 0 \\ x_1 + x_2 + x_3 &= 3 \\ x_1 - 3x_2 &= -2 \end{aligned}$$

$$\text{B.11 } \begin{aligned} 2x_1 + x_2 + x_3 &= 7 \\ 4x_2 - 5x_3 &= -7 \\ x_1 - 2x_2 + 4x_3 &= 9 \end{aligned}$$

$$\text{B.12 } \begin{aligned} 2x_1 + x_2 - 3x_3 + x_4 &= 1 \\ x_1 + 2x_2 + 5x_3 - x_4 &= 7 \\ -x_1 + x_2 + x_3 + 4x_4 &= 5 \\ 2x_1 - 3x_2 + 2x_3 - 5x_4 &= -4 \end{aligned}$$

$$\text{B.13 } \begin{aligned} 3x_1 + x_2 + x_3 &= 8 \\ 2x_1 - x_2 - x_3 &= -3 \\ x_1 + 2x_2 - x_3 &= 2 \end{aligned}$$

$$\text{B.14 } \begin{aligned} x_1 + x_2 - x_3 &= 2 \\ 2x_1 - x_2 + x_3 &= 4 \\ -x_1 + 2x_2 + 3x_3 &= 3 \end{aligned}$$

$$\text{B.15 } \begin{aligned} -x_1 + x_2 - x_3 &= -2 \\ -2x_1 + x_2 + 2x_3 &= 6 \\ x_1 + x_2 + x_3 &= 6 \end{aligned}$$

$$\text{B.16 } \begin{aligned} -x_1 + 2x_2 + 3x_3 &= 4 \\ 2x_1 - x_2 - 2x_3 &= -1 \\ x_1 - 3x_2 + 4x_3 &= 2 \end{aligned}$$

$$\begin{aligned} \text{B.17 } x_1 + x_2 + x_3 + x_4 &= 2 \\ 2x_1 + x_2 - x_3 + x_4 &= 2 \\ -x_1 + 2x_2 + 3x_3 + x_4 &= 1 \\ 3x_1 + 2x_2 - 2x_3 - x_4 &= 8 \end{aligned}$$

$$\begin{aligned} \text{B.18 } x_1 + x_2 + x_3 + x_4 &= -1 \\ 2x_1 - x_2 + x_3 - 2x_4 &= 8 \\ 3x_1 + 2x_2 + 2x_3 + 2x_4 &= 4 \\ -x_1 - x_2 + 2x_3 - x_4 &= -2 \end{aligned}$$

Check if the following systems of equations are consistent. If they are, calculate their general solution.

$$\begin{aligned} \text{B.19 } 3x_1 + x_2 + 5x_3 + 2x_4 &= 2 \\ 2x_1 - 2x_2 + 4x_3 &= 2 \\ 2x_1 + 2x_2 + 3x_3 + 2x_4 &= 1 \\ x_1 + 3x_2 + x_3 + 2x_4 &= 0 \end{aligned}$$

$$\begin{aligned} \text{B.20 } x_1 + x_2 + x_3 + x_4 &= 10 \\ -x_1 + x_2 - x_3 + x_4 &= 2 \\ 2x_1 - 3x_2 + 2x_3 - 2x_4 &= -6 \end{aligned}$$

$$\begin{aligned} \text{B.21 } x_2 + 2x_3 + x_4 &= -2 \\ x_1 - 2x_2 - x_3 - x_4 &= 1 \\ x_1 - 2x_2 - 3x_3 + x_4 &= 1 \end{aligned}$$

$$\begin{aligned} \text{B.22 } x_1 + x_2 + x_3 + x_4 &= 0 \\ 2x_1 + x_2 - 2x_3 - x_4 &= 6 \\ 3x_1 + 2x_2 + x_3 + 2x_4 &= 2 \end{aligned}$$

$$\begin{aligned} \text{B.23 } x_1 + x_2 + x_3 + 3x_4 - x_5 &= 5 \\ 2x_1 - x_2 + x_3 - x_4 + 3x_5 &= 4 \\ -x_1 + 2x_2 - x_3 + 3x_4 - 2x_5 &= 1 \end{aligned}$$

$$\begin{aligned} \text{B.24 } 2x_1 - x_2 + x_3 + x_4 - x_5 &= 2 \\ -x_1 + x_2 - x_3 - x_4 + x_5 &= -1 \\ 4x_1 + 2x_2 + 3x_3 + 2x_4 - x_5 &= 20 \end{aligned}$$

$$\begin{aligned} \text{B.25 } 3x_1 + 3x_2 + 2x_3 + x_4 &= 19 \\ 2x_1 - x_2 + x_3 + x_4 - x_5 &= 2 \\ 4x_1 + 2x_2 + 3x_3 + 2x_4 - x_5 &= 20 \end{aligned}$$

$$\begin{aligned} \text{B.26 } x_1 + x_2 + 2x_4 - x_5 &= 5 \\ x_1 + x_2 + x_3 + 3x_4 - x_5 &= 5 \\ 2x_1 - x_2 + x_3 - x_4 + 3x_5 &= 4 \\ -x_1 + 2x_2 - x_3 + 3x_4 - 2x_5 &= 1 \end{aligned}$$

$$\begin{aligned} \text{B.27 } x_2 + 2x_3 + x_4 + 3x_5 + 2x_6 &= 9 \\ -x_1 + 5x_2 + 2x_3 + x_4 + 2x_5 + x_7 &= 10 \\ 5x_1 - 3x_2 + 8x_3 + 6x_4 + 3x_5 - 2x_8 &= 17 \\ 2x_1 - x_2 + x_4 + 5x_5 - 2x_8 &= 5 \end{aligned}$$

Check the linear independence of the following set of vectors.

$$\text{B.28 } \mathbf{a}^{(1)} = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}, \mathbf{a}^{(2)} = \begin{bmatrix} -3 \\ -4 \\ 1 \end{bmatrix}, \mathbf{a}^{(3)} = \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix}, \mathbf{a}^{(4)} = \begin{bmatrix} 4 \\ 0 \\ 1 \end{bmatrix}$$

$$\text{B.29 } \mathbf{a}^{(1)} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}, \mathbf{a}^{(2)} = \begin{bmatrix} -2 \\ 1 \\ 0 \\ 1 \\ -1 \end{bmatrix}, \mathbf{a}^{(3)} = \begin{bmatrix} 4 \\ 0 \\ -3 \\ 2 \\ 1 \end{bmatrix}$$

Find eigenvalues for the following matrices.

$$\text{B.30 } \begin{bmatrix} 1 & 2 \\ 2 & 5 \end{bmatrix} \quad \text{B.31 } \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix}$$

$$\text{B.32 } \begin{bmatrix} 1 & 1 & 0 \\ 1 & 4 & 0 \\ 0 & 0 & 5 \end{bmatrix} \quad \text{B.33 } \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

$$\text{B.34 } \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 5 \end{bmatrix}$$

Appendix C A Numerical Method for Solution of Nonlinear Equations

Nonlinear equations are encountered in many fields of engineering. In design optimization, such equations arise when we write the necessary conditions of optimality for unconstrained or constrained problems, discussed in Chapter 4. Roots of that nonlinear set of equations are the candidate minimum designs. Thus the problem of finding *roots of nonlinear equations* must be treated.

The analytical solution of nonlinear equations is almost impossible except in very simple cases where elimination of variables can be carried out. Therefore, numerical methods and digital computers must be used to find the solutions of such systems. In this appendix, we describe a basic numerical method known as the *Newton-Raphson method for finding roots of nonlinear equations*. Many variations of the method as well as other methods are available in the literature (Atkinson, 1978). The method *can find only one root at a time* depending on the initial estimate for the root. Therefore, *several widely separated starting points must be tried* to find different roots. *The method may also fail to converge unless the starting point is in some neighbourhood of the solution. If it fails, a different starting point should be tried.*

We shall describe a basic algorithm that can be coded into a computer program. However, most computer center libraries have several programs for solving a system of nonlinear equations. These programs can be used directly. Therefore, the possibility of utilizing existing programs must be investigated before attempting to code the algorithm.

C.1 Single Nonlinear Equation

To develop the method, let us first consider the problem of finding roots of a general nonlinear equation

$$F(x) = 0 \tag{C.1}$$

where $F(x)$ is a nonlinear function of an independent variable x . A simple way of finding roots is to graph $F(x)$ versus x . Then, the points where the function crosses the x axis are the roots of $F(x) = 0$.

The second method developed by Newton-Raphson is an iterative numerical procedure in which we start with an initial estimate for a root. For the estimate, the function $F(x)$ will generally not have zero value, i.e., the initial estimate is usually not a root of $F(x) = 0$. Therefore, we try to improve the estimate until a root is found. This process, requiring *several cycles (or iterations)* before the root is found, can be described by the following equation:

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}, \quad k = 0, 1, 2, \dots \quad (\text{C.2})$$

where superscript k is the iteration number, $x^{(0)}$ is an initial estimate (starting point), and $\Delta x^{(k)}$ is a change in the estimate at the k th iteration. The iterative process is continued until $F(x)$ is reduced to zero or a small acceptable value, say δ (i.e., the stopping criterion is $|F| \leq \delta$). Thus the Newton-Raphson method is reduced to somehow computing $\Delta x^{(k)}$ at each iteration of the process. *Note that the iteration concept described by Eq. (C.2) is very general because many other numerical methods are based on the same formula.*

To develop an expression for $\Delta x^{(k)}$, we use a linear (first-order) Taylor series expansion for the function $F(x)$ about the current estimate $x^{(k)}$. Therefore, from Eq. (4.7), we have

$$F(x^{(k)} + \Delta x^{(k)}) \approx F(x^{(k)}) + \frac{dF(x^{(k)})}{dx} \Delta x^{(k)} \quad (\text{C.3})$$

In Eq. (C.3), $x^{(k)}$ is the current estimate for the root, $F(x^{(k)})$ is the current value of the function $F(x)$, and $dF(x^{(k)})/dx$ is the current slope (gradient) of $F(x)$. Our objective is to find $\Delta x^{(k)}$ (an improvement in the current estimate) such that the equation $F(x) = 0$ is satisfied at the new estimate, i.e., $F(x^{(k)} + \Delta x^{(k)}) = 0$. Therefore, setting Eq. (C.3) to zero, $\Delta x^{(k)}$ is given as

$$\Delta x^{(k)} = \frac{-F(x^{(k)})}{(dF/dx)} \quad (\text{C.4a})$$

For each k , $\Delta x^{(k)}$ is calculated from Eq. (C.4) and a new estimate for the root is obtained from Eq. (C.2). The process is repeated until $F(x)$ goes to zero.

The Newton-Raphson method also has a simple *geometrical representation*. To see this, let us consider the graph of a function $F(x)$ versus x shown in Fig. C-1. Point A on the curve represents the current value of the function $F(x)$, and point B on the x -axis represents the current estimate $x^{(k)}$ for the root of $F(x) = 0$. Point x^* is the root of $F(x) = 0$. The objective of the method is to reach point x^* . The method proceeds by following the tangent to the curve $F(x)$ at point A. Intersection of the tangent with the x -axis gives point C, $x^{(k+1)}$ from where the process is repeated.

The geometry of Fig. C-1 can be used to derive Eq. (C.4a). To do this, we consider the triangle ABC:

$$\frac{F(x^{(k)})}{\Delta x^{(k)}} = \tan \theta = -\frac{dF(x^{(k)})}{dx} \quad (\text{C.4b})$$

or

$$\Delta x^{(k)} = -\frac{F(x^{(k)})}{(dF/dx)} \quad (\text{C.4c})$$

which is same as Eq. (C.4a).

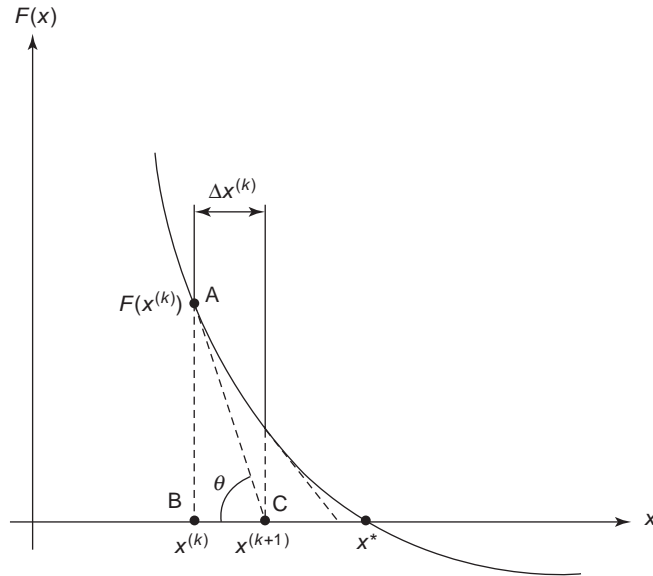


FIGURE C-1 Graphical representation of the Newton-Raphson method for finding roots of $F(x) = 0$.

Steps of the iterative Newton-Raphson method are summarized as follows:

Step 1. Select a starting point $x^{(0)}$ and the parameter δ for stopping the iterative process. Set the iteration counter $k = 0$.

Step 2. Calculate the function F at the current estimate $x^{(k)}$. Check for convergence; if $|F(x^{(k)})| \leq \delta$, then stop the iterative process and accept $x^{(k)}$ as a root of $F(x) = 0$. Otherwise continue.

Step 3. Calculate the derivative of the function dF/dx at the current estimate $x^{(k)}$.

Step 4. Calculate $\Delta x^{(k)} = -F(x^{(k)})/(dF/dx)$.

Step 5. Update the estimate for the root as $x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$.

Step 6. Set $k = k + 1$ and go to Step 2.

We demonstrate the procedure in the following example.

EXAMPLE C.1 Roots of a Nonlinear Equation by the Newton-Raphson Method

Find a root of the following equation using the Newton-Raphson method starting from the point $x^{(0)} = 1.0$:

$$F(x) = \frac{2x}{3} - \sin x = 0 \quad (\text{a})$$

Solution. The derivative of $F(x)$ for use in Eq. (C.4) is given as

$$dF/dx = \frac{2}{3} - \cos x \quad (\text{b})$$

We follow the steps of the foregoing algorithm:

1. $x^{(0)} = 1.0$; let $\delta = 0.001$; $k = 0$.
2. $F(x^{(0)}) = \frac{2}{3}(1) - \sin(1) = -0.1750$.
 $|F| = 0.1750 > 0.001$, so $x^{(0)} = 1.0$ is not a root.
3. $dF/dx = \frac{2}{3} - \cos(1) = 0.1264$.
4. $\Delta x^{(0)} = \frac{-F(x^{(0)})}{dF/dx} = -\frac{(-0.1750)}{(0.1264)} = 1.3830$.
5. $x^{(0+1)} = x^{(0)} + \Delta x^{(0)}$; or $x^{(1)} = 1.0 + 1.3830 = 2.3830$.
6. $k = 0 + 1 = 1$; go to Step 2.

Results of various iterations of the preceding procedure are summarized in Table C-1. At the fourth iteration $|F(x^{(4)})| = 0.00013 < 0.001$ satisfying the specified stopping criteria. Therefore, $x^{(4)} = 1.496$ is taken as an estimate of the root for the given function. To find other roots, we must repeat the preceding process from another starting point.

TABLE C-1 Newton-Raphson Iterations for Example C.1: $F(x) = 2x/3 - \sin x$

k	$x^{(k)}$	$F(x^{(k)})$	$dF(x^{(k)})/dx$	$\Delta x^{(k)}$
0	1.000	-0.175	0.1264	1.383
1	2.383	0.900	0.1393	-0.646
2	1.737	0.172	0.832	-0.207
3	1.530	0.021	0.626	-0.034
4	1.496	0.00013	—	—

C.2 Multiple Nonlinear Equations

The foregoing Newton-Raphson procedure can be generalized for the case of n nonlinear equations in n unknowns. We first derive the procedure and then summarize it in a step-by-step algorithm. The reader who is not interested in the derivation can go directly to the algorithm and the example problem.

A set of nonlinear equations can be written in vector form as

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \quad (\text{C.5})$$

where \mathbf{F} and \mathbf{x} are both n -dimensional vectors. In the iterative procedure, we start with an estimate $\mathbf{x}^{(0)}$ for the root of Eq. (C.5). Just as before, this estimate is improved based on the vector form of Eq. (C.2) as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta \mathbf{x}^{(k)}, \quad k = 0, 1, 2, \dots \quad (\text{C.6})$$

where $\mathbf{x}^{(0)}$ is an initial estimate, k is an iteration number, and $\Delta \mathbf{x}^{(k)}$ is a vector of changes in the estimate $\mathbf{x}^{(k)}$. The iterative procedure is continued until $\mathbf{x}^{(k)}$ satisfies Eq. (C.5). Since numerical computations are not exact, we need a criterion for judging when a root is found. A common method is to calculate the length of the vector $\mathbf{F}(\mathbf{x})$ as

$$\|\mathbf{F}(\mathbf{x})\| = \left(\sum_{i=1}^n \{F_i(\mathbf{x})\}^2 \right)^{1/2} \quad (\text{C.7})$$

and accept \mathbf{x}^* as a root of $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ if

$$\|\mathbf{F}(\mathbf{x}^*)\| \leq \delta \quad (\text{C.8})$$

where $\delta > 0$ is some small specified number. Another stopping criterion might be to require the largest component of $\mathbf{F}(\mathbf{x})$ to satisfy the stopping criterion, i.e., $|F_i|_{\max} \leq 0$.

Now our task is to develop a formula for $\Delta\mathbf{x}^{(k)}$ so that Eq. (C.6) may be used to improve the estimate for the root of Eq. (C.5). To do this, we follow the procedure used for the case of one equation in one variable, i.e., we write a linear Taylor series expansion of each function. To derive the procedure, we consider the case of two equations in two unknowns in Eq. (C.5) and then generalize the result to the case of n equations in n unknowns. Equation (C.5) for the case of $n = 2$ is given as

$$\begin{aligned} F_1(x_1, x_2) &= 0 \\ F_2(x_1, x_2) &= 0 \end{aligned} \quad (\text{a})$$

and the iterative equation (C.6) becomes

$$\begin{aligned} x_1^{(k+1)} &= x_1^{(k)} + \Delta x_1^{(k)} \\ x_2^{(k+1)} &= x_2^{(k)} + \Delta x_2^{(k)} \end{aligned} \quad (\text{b})$$

To obtain expression for $\Delta x_1^{(k)}$ and $\Delta x_2^{(k)}$, we write linear Taylor series expansions for the functions F_1 and F_2 in Eq. (a) about the current estimate $x^{(k)}$ and, as before, set them to zero:

$$\begin{aligned} F_1(x_1^{(k)}, x_2^{(k)}) + \frac{\partial F_1}{\partial x_1} \Delta x_1^{(k)} + \frac{\partial F_1}{\partial x_2} \Delta x_2^{(k)} &= 0 \\ F_2(x_1^{(k)}, x_2^{(k)}) + \frac{\partial F_2}{\partial x_1} \Delta x_1^{(k)} + \frac{\partial F_2}{\partial x_2} \Delta x_2^{(k)} &= 0 \end{aligned} \quad (\text{c})$$

where the partial derivatives $\partial F_i / \partial x_j$ are evaluated at the current estimate $x_1^{(k)}$ and $x_2^{(k)}$. Equation (c) shows two linear equations in two unknowns $\Delta x_1^{(k)}$ and $\Delta x_2^{(k)}$; all other quantities are known. Therefore, they are solved for $\Delta x_1^{(k)}$ and $\Delta x_2^{(k)}$, and Eq. (b) is used to update the estimate for the root. The process is repeated until convergence is achieved.

Equation (c) can be written in matrix form as

$$\begin{bmatrix} F_1^{(k)} \\ F_2^{(k)} \end{bmatrix} + \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} \Delta x_1^{(k)} \\ \Delta x_2^{(k)} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (\text{d})$$

where $F_1^{(k)} = F_1(x_1^{(k)}, x_2^{(k)})$ and $F_2^{(k)} = F_2(x_1^{(k)}, x_2^{(k)})$ are the function values at the current estimate $\mathbf{x}^{(k)}$. Or, we can write the equation compactly as

$$\mathbf{F}^{(k)} + \mathbf{J}[\Delta \mathbf{x}^{(k)}] = \mathbf{0} \quad (\text{C.9})$$

where the vector $\mathbf{F}^{(k)}$, matrix \mathbf{J} , and vector $\Delta \mathbf{x}^{(k)}$ are easily identified from Eq. (d). The matrix \mathbf{J} in Eq. (C.9) is usually called the *Jacobian* of the system of equations.

Equation (C.9) allows us to generalize the Newton-Raphson method for n equations in n unknowns. In that case $\mathbf{F}^{(k)}$ and $\Delta \mathbf{x}^{(k)}$ become n -dimensional vectors and \mathbf{J} becomes an $n \times n$ matrix of partial derivatives defined as

$$\mathbf{J} = \begin{bmatrix} \frac{\partial F_i}{\partial x_j} \end{bmatrix}; \quad i = 1 \text{ to } n; \quad j = 1 \text{ to } n \quad (\text{C.10})$$

Using the procedure for calculating the Jacobian for two equations in two unknowns identified in Eq. (d), we observe that the i th row of \mathbf{J} in Eq. (C.10) is obtained by differentiating the function $F_i(\mathbf{x})$ with respect to all the variables. That is, the first row is obtained by differentiating F_1 with respect to x_1, x_2, \dots, x_n , the second row by differentiating F_2 , and so on. Note that the i th row of the Jacobian in Eq. (C.10) can be also considered as a transpose of the gradient vector of $F_i(\mathbf{x})$, i.e., ∇F_i^T .

If the inverse of the matrix \mathbf{J} can be calculated, then an improvement to the estimate for the root is obtained from Eq. (C.9) as

$$\Delta \mathbf{x}^{(k)} = -\mathbf{J}^{-1} \mathbf{F}^{(k)} \quad (\text{C.11})$$

In numerical calculations, however, it is inefficient to invert matrices. Therefore, it is recommended that $\Delta \mathbf{x}^{(k)}$ be computed by solving the following linear system of equations obtained from Eq. (C.9) as

$$\mathbf{J}[\Delta \mathbf{x}^{(k)}] = -\mathbf{F}^{(k)} \quad (\text{C.12})$$

The *Newton-Raphson algorithm* is then summarized as follows:

- Step 1.* Select a starting point $\mathbf{x}^{(0)}$ and the parameter δ for stopping the iterative process. Set the iteration counter $k = 0$.
- Step 2.* Calculate the functions in \mathbf{F} at the current estimate $\mathbf{x}^{(k)}$. Check for convergence; if $\|\mathbf{F}^{(k)}\| \leq \delta$, then stop the iterative process and accept $\mathbf{x}^{(k)}$ as a root of the equation $\mathbf{F}(\mathbf{x}) = 0$. Otherwise, continue.
- Step 3.* Calculate the Jacobian matrix \mathbf{J} of partial derivatives at the current estimate $\mathbf{x}^{(k)}$ as in Eq. (C.10), i.e., calculate $[\partial F_i / \partial x_j]$; $i = 1$ to n ; $j = 1$ to n .
- Step 4.* Calculate $\Delta \mathbf{x}^{(k)}$ from Eq. (C.12).
- Step 5.* Update the estimate for the root using Eq. (C.6).
- Step 6.* Set $k = k + 1$ and go to Step 2.

Note that since the system of equations $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ is nonlinear, it has in general many roots. *The Newton-Raphson algorithm finds only one root at a time.* The method converges to a root depending on the starting estimate $\mathbf{x}^{(0)}$. To find other roots we should restart the algorithm by selecting a different initial estimate $\mathbf{x}^{(0)}$. *Note also that the method will not work if the Jacobian \mathbf{J} is singular at any iteration* since its inverse cannot be calculated. In addition, the method may fail to converge even if Jacobian \mathbf{J} is nonsingular at all iterations. Several modifications of the basic Newton-Raphson method have been developed to make the method stable and convergent. These extensions, however, are beyond the scope of the present text.

EXAMPLE C.2 Roots of Nonlinear Equations by the Newton-Raphson Method

Find a root of the following 2×2 system of nonlinear equations using the Newton-Raphson procedure:

$$F_1(x_1, x_2) = 1.0 - \frac{(4.0E+06)}{x_1^2 x_2} = 0 \quad (a)$$

$$F_2(x_1, x_2) = 250.0 - \frac{(4.0E+06)}{x_1 x_2^2} = 0 \quad (b)$$

Solution. To use the Newton-Raphson algorithm, we need to compute the Jacobian for the system of Eqs. (a) and (b). Using the definition given in Eq. (C.10), the Jacobian is given as

$$\mathbf{J} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} \end{bmatrix} = (4.0E+06) \begin{bmatrix} \frac{2}{x_1^3 x_2} & \frac{1}{x_2^2 x_1^2} \\ \frac{1}{x_1^2 x_2^2} & \frac{2}{x_1 x_2^3} \end{bmatrix} \quad (c)$$

We use the steps of the algorithm as follows:

1. Let $\mathbf{x}^{(0)} = (500.0, 1.0)$ and $\delta = 0.10$; set $k = 0$.
2. Calculate the function values from Eqs. (a) and (b)

$$F_1 = 1.0 - (4.0E+06)/(500.0 \times 500.0 \times 1.0) = -15$$

$$F_2 = 250.0 - (4.0E+06)/(500.0 \times 1.0 \times 1.0) = -7750$$

$\|\mathbf{F}\| = \sqrt{15^2 + 7750^2} = 7750 > 0.10$ so $\mathbf{x}^{(0)}$ is not a root; continue the iterative process.

3. Jacobian matrix is calculated using Eq. (c):

$$\begin{aligned} \mathbf{J} &= (4.0E+06) \begin{bmatrix} \frac{2}{(500)^3 (1.0)} & \frac{1}{(500)^2 (1.0)^2} \\ \frac{1}{(500)^2 (1.0)^2} & \frac{2}{(500)(1.0)^3} \end{bmatrix} \\ &= \begin{bmatrix} \frac{8}{125} & 16 \\ 16 & 16,000 \end{bmatrix} \end{aligned} \quad (d)$$

4. Equation (C.9) or (C.12) defines the following linear system of equations for $\Delta x_1^{(0)}$ and $\Delta x_2^{(0)}$:

$$\begin{bmatrix} \frac{8}{125} & 16 \\ 16 & 16,000 \end{bmatrix} \begin{bmatrix} \Delta x_1^{(0)} \\ \Delta x_2^{(0)} \end{bmatrix} = \begin{bmatrix} 15 \\ 7750 \end{bmatrix} \quad (e)$$

Solving the two equations by the elimination process, we get $\Delta x_1^{(0)} = 151.0$, $\Delta x_2^{(0)} = 0.330$.

For this problem it is possible to invert the Jacobian matrix \mathbf{J} and use Eq. (C.11) for calculating $\Delta x_1^{(0)}$ and $\Delta x_2^{(0)}$. \mathbf{J}^{-1} is calculated using the cofactors approach (Appendix B, Section B.3) as

$$\begin{aligned}\mathbf{J}^{-1} &= \frac{1}{|\mathbf{J}|} [\text{cofac}(\mathbf{J})]^T \\ &= \frac{1}{768} \begin{bmatrix} 16,000 & -16 \\ -16 & \frac{8}{125} \end{bmatrix}\end{aligned}$$

Using Eq. (C.11), we obtain the same values for $\Delta x_1^{(0)}$ and $\Delta x_2^{(0)}$ as before.

5. Update the estimate for the root using Eq. (C.6) or Eqs. (b)

$$x_1^{(0+1)} = x_1^{(0)} + \Delta x_1^{(0)} = 500.0 + 151.0 = 651.0$$

$$x_2^{(0+1)} = x_2^{(0)} + \Delta x_2^{(0)} = 1.00 + 0.33 = 1.33$$

6. Set $k = 0 + 1 = 1$, and go to Step 2.

Results from various iterations of the Newton-Raphson algorithm are summarized in Table C-2. The table is generated by repeating the six steps of the algorithm. At the seventh iteration $\|\mathbf{F}^{(7)}\| = 0.05 < 0.10$. Thus $\mathbf{x}^{(7)} = (1000.3, 3.9995)$ is considered as a root for the system of Eqs. (a) and (b) satisfying the desired accuracy. The exact root is $\mathbf{x}^* = (1000.0, 4.0)$.

TABLE C-2 Newton-Raphson Iterations for Example C.2

k	$x_1^{(k)}$	$x_2^{(k)}$	$\ \mathbf{F}^{(k)}\ $
0	500.0	1.0000	7750.00
1	651.0	1.3300	3206.00
2	822.0	1.7700	1291.00
3	976.0	2.3500	489.20
4	1047.0	3.0490	161.30
5	1025.0	3.6770	38.50
6	1003.0	3.9630	3.98
7	1000.3	3.9995	0.05

Exercises for Appendix C

Find roots of the following equations using the Newton-Raphson method and $\delta = 0.001$.

C.1 $F(x) = 3x - e^x = 0$ starting from $x = 0$.

C.2 $F(x) = \sin x = 0$ starting from $x = 10$.

C.3 $F(x) = \cos x = 0$ starting from $x = 2$.

C.4 $F(x) = \frac{2x}{3} - \sin x = 0$ starting from $x = -4$.

Complete two iterations of the Newton-Raphson method for the following systems of nonlinear equations.

C.5 $F_1(\mathbf{x}) = 1 - \frac{10}{x_1^2 x_2} = 0$

$F_2(\mathbf{x}) = 5 - \frac{10}{x_1 x_2^2} = 0$; starting point (4, 1)

C.6 $F_1(\mathbf{x}) = 5 - \frac{1}{8}x_1 x_2 - \frac{1}{4x_1^2}x_2^2 = 0$

$F_2(\mathbf{x}) = -\frac{1}{16}x_1^2 + \frac{1}{2x_1}x_2 = 0$; starting point (10, 10)

C.7 $F_1(\mathbf{x}) = 3x_1^2 + 12x_2^2 + 10x_1 = 0$

$F_2(\mathbf{x}) = 24x_1 x_2 + 4x_2 + 3 = 0$; starting point (-5, 0)

Find all roots of the following systems of nonlinear equations using a computer program.

C.8 Exercise C.5

C.9 Exercise C.6

C.10 Exercise C.7

Appendix D Sample Computer Programs

This appendix contains the *listing of some computer programs based on the algorithms for numerical methods of unconstrained optimization given in Chapters 8 and 9*. The objective is to educate the student on how to transform a step-by-step numerical algorithm into a computer program. Note that the given computer programs are not claimed to be the most efficient ones. The key idea is to emphasize the essential numerical aspects of the algorithms in a simple and straightforward way. A beginner in the numerical techniques of optimization is expected to experiment with the computer programs and get a feel for the various methods by solving some numerical examples. Thus, a black box usage of the computer programs given in this appendix is discouraged.

D.1 Equal Interval Search

As discussed in Chapter 8, equal interval search is the simplest method of one-dimensional minimization. A computer program based on it is given in Fig. D-1. It is assumed that the one-dimensional function is unimodal and continuous, and has a negative slope in the interval of interest. The initial step length (δ) and line search accuracy (ϵ) must be specified in the main program. The subroutine EQUAL is called from the main program to perform a line search by equal interval search. The three major tasks to be accomplished in the subroutine EQUAL are: (1) to establish initial step length δ such that $f(0) > f(\delta)$, (2) to establish the initial interval of uncertainty, (α_l, α_u) , and (3) to reduce the interval of uncertainty such that $(\alpha_l - \alpha_u) \leq \epsilon$.

The subroutine EQUAL calls the subroutine FUNCT to evaluate the value of the one-dimensional function at various trial steps. The subroutine FUNCT is supplied by the user. As an example, $f(\alpha) = 2 - 4\alpha + e^\alpha$ is chosen as the one-dimensional minimization function. The listing of the program in Fig. D-1 is self-explanatory. In the subroutine EQUAL, the following notation is used:

AL = lower limit on α ; α_l
AU = upper limit on α ; α_u
FL = function value at α_l ; $f(\alpha_l)$
FU = function value at α_u ; $f(\alpha_u)$
AA = intermediate point α_a
FA = function value at α_a ; $f(\alpha_a)$

```

C      MAIN PROGRAM FOR EQUAL INTERVAL SEARCH

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      DELTA = 5.0D-2
      EPSLON = 1.0D-3
      NCOUNT = 0
      F      = 0.0D0
      ALFA   = 0.0D0

C
C      TO PERFORM LINE SEARCH CALL SUBROUTINE EQUAL
C
      CALL EQUAL(ALFA,DELTA,EPSLON,F,NCOUNT)
      WRITE(*,10) ' MINIMUM =', ALFA
      WRITE(*,10) ' MINIMUM FUNCTION VALUE =', F
      WRITE(*,*) 'NO. OF FUNCTION EVALUATIONS =', NCOUNT
10     FORMAT(A,1PE14.5)

      STOP
      END

      SUBROUTINE EQUAL(ALFA,DELTA,EPSLON,F,NCOUNT)
C      -----
C      THIS SUBROUTINE IMPLEMENTS EQUAL INTERVAL SEARCH
C      ALFA = OPTIMUM VALUE ON RETURN
C      DELTA = INITIAL STEP LENGTH
C      EPSLON= CONVERGENCE PARAMETER
C      F      = OPTIMUM VALUE OF THE FUNCTION ON RETURN
C      NCOUNT= NUMBER OF FUNCTION EVALUATIONS ON RETURN
C      -----

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

C
C      ESTABLISH INITIAL DELTA
C
      AL = 0.0D0
      CALL FUNCT(AL,FL,NCOUNT)
10     CONTINUE
      AA = DELTA
      CALL FUNCT(AA,FA,NCOUNT)
      IF (FA .GT. FL) THEN
          DELTA = DELTA * 0.1D0
          GO TO 10
      END IF

```

FIGURE D-1 Program for equal interval search.


```

C
C   ESTABLISH INITIAL INTERVAL OF UNCERTAINTY
C
20  CONTINUE
    AU = AA + DELTA
    CALL FUNCT(AU,FU,NCOUNT)
    IF (FA .GT. FU) THEN
        AL = AA
        AA = AU
        FL = FA
        FA = FU
        GO TO 20
    END IF

C
C   REFINE THE INTERVAL OF UNCERTAINTY FURTHER
C
30  CONTINUE
    IF ((AU - AL) .LE. EPSLON) GO TO 50
    DELTA = DELTA * 0.1D0
    AA = AL
    FA = FL
40  CONTINUE
    AU = AA + DELTA
    CALL FUNCT(AU,FU,NCOUNT)
    IF (FA .GT. FU) THEN
        AL = AA
        AA = AU
        FL = FA
        FA = FU
        GO TO 40
    END IF
    GO TO 30

C
C   MINIMUM IS FOUND
C
50  ALFA = (AU + AL) * 0.5D0
    CALL FUNCT(ALFA,F,NCOUNT)

    RETURN
    END

```

FIGURE D-1 Continued

```

SUBROUTINE FUNCT(AL, F, NCOUNT)
C -----
C CALCULATES THE FUNCTION VALUE
C AL      = VALUE OF ALPHA, INPUT
C F       = FUNCTION VALUE ON RETURN
C NCOUNT = NUMBER OF CALLS FOR FUNCTION EVALUATION
C -----

IMPLICIT DOUBLE PRECISION (A-H, O-Z)

NCOUNT = NCOUNT + 1
C      F = 1.0D0 - 3.0D0 * AL + DEXP(2.0D0 * AL)
      F = 18.5D0*AL**2-85.0D0*AL-13.5D0

RETURN
END

```

FIGURE D-1 Continued

D.2 Golden Section Search

Golden section search is considered to be one of the efficient methods requiring only function values. The subroutine GOLD, given in Fig. D-2, implements the golden section search algorithm given in Chapter 8 and is called from the main program given in Fig. D-1; the call to subroutine EQUAL is replaced by a call to the subroutine GOLD. The initial step length and initial interval of uncertainty are established in GOLD, as in the subroutine EQUAL. The interval of uncertainty is reduced further to satisfy the line search accuracy by implementing Step 3 of the algorithm given in Chapter 8. The subroutine FUNCT is used to evaluate the function value at a trial step. The following notation is used in the subroutine GOLD: $AA = \alpha_a$, $AB = \alpha_b$, $AL = \alpha_l$, $AU = \alpha_u$, $FA = f(\alpha_a)$, $FB = f(\alpha_b)$, $FL = f(\alpha_l)$, $FU = f(\alpha_u)$, and GR = golden ratio, $(\sqrt{5} + 1)/2$.

D.3 Steepest Descent Method

The steepest descent method is the simplest of the gradient-based methods for unconstrained optimization. A computer program for the method is given in Fig. D-3. The basic steps in the algorithm are: (1) evaluate the gradient of the cost function at the current point, (2) evaluate an optimum step size along the negative gradient direction, and (3) update the design, check the convergence criterion, and if necessary repeat the preceding steps. The main program essentially follows these steps. The arrays declared in the main program must have dimensions of the design variable vector. Also, the initial data and starting point must be provided by the user. The cost function and its gradient must be provided in subroutines FUNCT and GRAD, respectively. The line search is performed in subroutine GOLDM by golden section search for a multivariate problem. As an example, $f(\mathbf{x}) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$ is chosen as the cost function.

```

SUBROUTINE GOLD(ALFA,DELTA,EPSLON,F,NCOUNT)
C -----
C THIS SUBROUTINE IMPLEMENTS GOLDEN SECTION SEARCH
C ALFA = OPTIMUM VALUE OF ALPHA ON RETURN
C DELTA = INITIAL STEP LENGTH
C EPSLON= CONVERGENCE PARAMETER
C F = OPTIMUM VALUE OF THE FUNCTION ON RETURN
C NCOUNT= NUMBER OF FUNCTION EVALUATIONS ON RETURN
C -----

IMPLICIT DOUBLE PRECISION(A-H,O-Z)

GR = 0.5D0 * SQRT(5.0D0) + 0.5D0

C ESTABLISH INITIAL DELTA
C
AL = 0.0D0
CALL FUNCT(AL,FL,NCOUNT)
10 CONTINUE
AA = DELTA
CALL FUNCT(AA,FA,NCOUNT)
IF (FA .GT. FL) THEN
    DELTA = DELTA * 0.1D0
    GO TO 10
END IF

C ESTABLISH INITIAL INTERVAL OF UNCERTAINTY
C
J = 0
20 CONTINUE
J = J + 1
AU = AA + DELTA * (GR ** J)
CALL FUNCT(AU,FU,NCOUNT)
IF (FA .GT. FU) THEN
    AL = AA
    AA = AU
    FL = FA
    FA = FU
    GO TO 20
END IF

C REFINE THE INTERVAL OF UNCERTAINTY FURTHER
C
AB = AL + (AU - AL) / GR
CALL FUNCT(AB,FB,NCOUNT)
30 CONTINUE

```

FIGURE D-2 Subroutine GOLD for golden section search.

```

      IF ((AU - AL) .LE. EPSLON) GO TO 80
C
C   IMPLEMENT STEPS 4 ,5 OR 6 OF THE ALGORITHM
C
      IF (FA - FB) 40, 60, 50
C
C   FA IS LESS THAN FB (STEP 4)
40   AU = AB
      FU = FB
      AB = AA
      FB = FA
      AA = AL + (AU - AL) * (1.0D0 - 1.0D0 / GR)
      CALL FUNCT(AA,FA,NCOUNT)
      GO TO 30
C
C   FA IS GREATER THAN FB (STEP 5)
C
50   AL = AA
      FL = FA
      AA = AB
      FA = FB
      AB = AL + (AU - AL) / GR
      CALL FUNCT(AB,FB,NCOUNT)
      GO TO 30
C
C   FA IS EQUAL TO FB (STEP 6)
C
60   AL = AA
      FL = FA
      AU = AB
      FU = FB
      AA = AL + (1.0D0 - 1.0D0 / GR) * (AU - AL)
      CALL FUNCT(AA,FA,NCOUNT)
      AB = AL + (AU - AL) / GR
      CALL FUNCT(AB,FB,NCOUNT)
      GO TO 30
C
C   MINIMUM IS FOUND
C
80   ALFA = (AU + AL) * 0.5D0
      CALL FUNCT(ALFA,F,NCOUNT)
      RETURN
      END

```

FIGURE D-2 Continued

```

C      THE MAIN PROGRAM FOR STEEPEST DESCENT METHOD
C      -----
C      DELTA = INITIAL STEP LENGTH FOR LINE SEARCH
C      EPSLON= LINE SEARCH ACCURACY
C      EPSL  = STOPPING CRITERION FOR STEEPEST DESCENT METHOD
C      NCOUNT= NO. OF FUNCTION EVALUATIONS
C      NDV   = NO. OF DESIGN VARIABLES
C      NOC   = NO. OF CYCLES OF THE METHOD
C      X     = DESIGN VARIABLE VECTOR
C      D     = DIRECTION VECTOR
C      G     = GRADIENT VECTOR
C      WK    = WORK ARRAY USED FOR TEMPORARY STORAGE
C      -----

      IMPLICIT DOUBLE PRECISION (A-H, O-Z)
      DIMENSION X(4), D(4), G(4), WK(4)

C
C      DEFINE INITIAL DATA
C
      DELTA = 5.0D-2
      EPSLON= 1.0D-4
      EPSL  = 5.0D-3
      NCOUNT= 0
      NDV   = 3
      NOC   = 100

C
C      STARTING VALUES OF THE DESIGN VARIABLES
C
      X(1)=2.0D0
      X(2)=4.0D0
      X(3)=10.0D0

      CALL GRAD(X,G,NDV)
      WRITE(*,10)
10  FORMAT(' NO.      COST FUNCT      STEP SIZE',
&        '  NORM OF GRAD  ')
      DO 20 K = 1, NOC
          CALL SCALE (G,D,-1.0D0,NDV)
          CALL GOLDM(X,D,WK,ALFA,DELTA,EPSLON,F,NCOUNT,NDV)
          CALL SCALE(D,D,ALFA,NDV)
          CALL PRINT(K,X,ALFA,G,F,NDV)
          CALL ADD(X,D,X,NDV)
          CALL GRAD(X,G,NDV)
          IF(TNORM(G,NDV) .LE. EPSL) GO TO 30
20  CONTINUE

```

FIGURE D-3 Computer program for steepest descent method.

```

WRITE(*,*)
WRITE(*,*) ' LIMIT ON NO. OF CYCLES HAS EXCEEDED '
WRITE(*,*) ' THE CURRENT DESIGN VARIABLES ARE: '
WRITE(*,*) X
CALL EXIT

30  WRITE(*,*)
    WRITE(*,*) 'THE OPTIMAL DESIGN VARIABLES ARE: '
    WRITE(*,40) X
40  FORMAT (3F15.6)

    CALL FUNCT(X,F,NCOUNT,NDV)
    WRITE(*,50) ' THE OPTIMUM COST FUNCTION VALUE IS :', F
50  FORMAT(A, F13.6)
    WRITE(*,*) 'TOTAL NO. OF FUNCTION EVALUATIONS ARE', NCOUNT

    STOP
    END

    SUBROUTINE GRAD(X,G,NDV)
C
C   CALCULATES THE GRADIENT OF F(X) IN VECTOR G
C
    IMPLICIT DOUBLE PRECISION (A-H, O-Z)
    DIMENSION X(NDV),G(NDV)

    G(1) = 2.0D0 * X(1) + 2.0D0 * X(2)
    G(2) = 2.0D0 * X(1) + 4.0D0 * X(2) + 2.0D0 * X(3)
    G(3) = 2.0D0 * X(2) + 4.0D0 * X(3)

    RETURN
    END

    SUBROUTINE SCALE(A,X,S,M)
C
C   MULTIPLIES VECTOR A(M) BY SCALAR S AND STORES IN X(M)
C
    IMPLICIT DOUBLE PRECISION (A-H, O-Z)
    DIMENSION A(M),X(M)

```

FIGURE D-3 Continued

```

        DO 10 I = 1, M
            X(I) = S * A(I)
10     CONTINUE

        RETURN
        END

        REAL*8 FUNCTION TNORM(X,N)
C
C     CALCULATES NORM OF VECTOR X(N)
C
        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
        DIMENSION X(N)

        SUM = 0.0D0
        DO 10 I = 1, N
            SUM = SUM + X(I) * X(I)
10     CONTINUE
        TNORM = DSQRT(SUM)

        RETURN
        END

        SUBROUTINE ADD(A,X,C,M)
C
C     ADDS VECTORS A(M) AND X(M) AND STORES IN C(M)
C
        IMPLICIT DOUBLE PRECISION (A-H, O-Z)
        DIMENSION A(M), X(M), C(M)

        DO 10 I = 1, M
            C(I) = A(I) + X(I)
10     CONTINUE

        RETURN
        END

```

FIGURE D-3 Continued

```

SUBROUTINE PRINT(I,X,ALFA,G,F,M)
C
C PRINTS THE OUTPUT
C
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
DIMENSION X(M),G(M)

WRITE(*,10) I, F, ALFA, TNORM(G,M)
10 FORMAT(I4, 3F15.6)

RETURN
END

SUBROUTINE FUNCT(X,F,NCOUNT,NDV)
C
C CALCULATES THE FUNCTION VALUE
C
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
DIMENSION X(NDV)

NCOUNT = NCOUNT + 1
F = X(1) ** 2 + 2.D0 * (X(2) **2) + 2.D0 * (X(3) ** 2)
& + 2.0D0 * X(1) * X(2) + 2.D0 * X(2) * X(3)

RETURN
END

SUBROUTINE UPDATE (XN,X,D,AL,NDV)
C
C UPDATES THE DESIGN VARIABLE VECTOR
C
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
DIMENSION XN(NDV), X(NDV), D(NDV)

DO 10 I = 1, NDV
XN(I) = X(I) + AL * D(I)
10 CONTINUE

RETURN
END

```

FIGURE D-3 Continued


```

SUBROUTINE GOLDM(X,D,XN,ALFA,DELTA,EPSLON,F,NCOUNT,NDV)
C -----
C IMPLEMENTS GOLDEN SECTION SEARCH FOR MULTIVARIATE PROBLEMS
C X      = CURRENT DESIGN POINT
C D      = DIRECTION VECTOR
C XN     = CURRENT DESIGN + TRIAL STEP * SEARCH DIRECTION
C ALFA   = OPTIMUM VALUE OF ALPHA ON RETURN
C DELTA  = INITIAL STEP LENGTH
C EPSLON= CONVERGENCE PARAMETER
C F      = OPTIMUM VALUE OF THE FUNCTION
C NCOUNT= NUMBER OF FUNCTION EVALUATIONS ON RETURN
C -----

IMPLICIT DOUBLE PRECISION (A-H, O-Z)
DIMENSION X(NDV), D(NDV), XN(NDV)

GR = 0.5D0 * DSQRT(5.0D0) + 0.5D0
DELTA1 = DELTA

C
C ESTABLISH INITIAL DELTA
C
AL = 0.0D0
CALL UPDATE(XN,X,D,AL,NDV)
CALL FUNCT(XN,FL,NCOUNT,NDV)
F = FL
10 CONTINUE
AA = DELTA1
CALL UPDATE(XN,X,D,AA,NDV)
CALL FUNCT(XN,FA,NCOUNT,NDV)
IF (FA .GT. FL) THEN
    DELTA1 = DELTA1 * 0.1D0
    GO TO 10
END IF

C
C ESTABLISH INITIAL INTERVAL OF UNCERTAINTY
C
J = 0
20 CONTINUE
J = J + 1
AU = AA + DELTA1 * (GR ** J)

CALL UPDATE(XN,X,D,AU,NDV)
CALL FUNCT(XN,FU,NCOUNT,NDV)
IF (FA .GT. FU) THEN
    AL = AU

```

FIGURE D-3 Continued

```

        AA = AU
        FL = FA
        FA = FU
        GO TO 20
    END IF
C
C   REFINED THE INTERVAL OF UNCERTAINTY FURTHER
C
    AB = AL + (AU - AL) / GR
    CALL UPDATE(XN,X,D,AB,NDV)
    CALL FUNCT(XN,FB,NCOUNT,NDV)
30   CONTINUE
    IF((AU-AL) .LE. EPSLON) GO TO 80
C
C   IMPLEMENT STEPS 4 ,5 OR 6 OF THE ALGORITHM
C
    IF (FA-FB) 40, 60, 50
C
C   FA IS LESS THAN FB (STEP 4)
C
40   AU = AB
    FU = FB
    AB = AA
    FB = FA
    AA = AL + (1.0D0 - 1.0D0 / GR) * (AU - AL)
    CALL UPDATE(XN,X,D,AA,NDV)
    CALL FUNCT(XN,FA,NCOUNT,NDV)
    GO TO 30
C
C   FA IS GREATER THAN FB (STEP 5)
C
50   AL = AA
    FL = FA
    AA = AB
    FA = FB
    AB = AL + (AU - AL) / GR
    CALL UPDATE(XN,X,D,AB,NDV)
    CALL FUNCT(XN,FB,NCOUNT,NDV)
    GO TO 30
C
C   FA IS EQUAL TO FB (STEP 6)
C

```

FIGURE D-3 Continued

```

60  AL = AA
    FL = FA
    AU = AB
    FU = FB
    AA = AL + (1.0D0 - 1.0D0 / GR) * (AU - AL)
    CALL UPDATE(XN,X,D,AA,NDV)
    CALL FUNCT(XN,FA,NCOUNT,NDV)
    AB = AL + (AU - AL) / GR
    CALL UPDATE(XN,X,D,AB,NDV)
    CALL FUNCT(XN,FB,NCOUNT,NDV)
    GO TO 30

C
C  MINIMUM IS FOUND
C
80  ALFA = (AU + AL) * 0.5D0

    RETURN
    END

```

FIGURE D-3 Continued

D.4 Modified Newton's Method

The modified Newton's method evaluates the gradient as well as the Hessian for the function and thus has a quadratic rate of convergence. Note that even though the method has a superior rate of convergence, it may fail to converge because of the singularity or indefiniteness of the Hessian matrix of the cost function. A program for the method is given in Fig. D-4. The cost function, gradient vector, and Hessian matrix are calculated in the subroutines FUNCT, GRAD, and HASN, respectively. As an example, $f(\mathbf{x}) = x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_2x_3$ is chosen as the cost function. The Newton direction is obtained by solving a system of linear equations in the subroutine SYSEQ. It is likely that the Newton direction may not be a descent direction in which the line search will fail to evaluate an appropriate step size. In such a case, the iterative loop is stopped and an appropriate message is printed. The main program for the modified Newton's method and the related subroutines are given in Fig. D-4.

```

C   THE MAIN PROGRAM FOR MODIFIED NEWTON'S METHOD
C   -----
C   DELTA = INITIAL STEP LENGTH FOR LINE SEARCH
C   EPSLON= LINE SEARCH ACCURACY
C   EPSL  = STOPPING CRITERION FOR MODIFIED NEWTON'S METHOD
C   NCOUNT= NO. OF FUNCTION EVALUATIONS
C   NDV   = NO. OF DESIGN VARIABLES
C   NOC   = NO. OF CYCLES OF THE METHOD
C   X     = DESIGN VARIABLE VECTOR
C   D     = DIRECTION VECTOR
C   G     = GRADIENT VECTOR
C   H     = HESSIAN MATRIX
C   WK    = WORK ARRAY USED FOR TEMPORARY STORAGE
C   -----

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(3), D(3), G(3), H(3,3), WK(3)

C
C   DEFINE INITIAL DATA
C
      DELTA = 5.0D-2
      EPSLON = 1.0D-4
      EPSL = 5.0D-3
      NCOUNT = 0
      NDV = 3
      NOC = 100

C
C   STARTING VALUES OF THE DESIGN VARIABLES
C
      X(1) = 2.0D0
      X(2) = 4.0D0
      X(3) = 10.0D0

      CALL GRAD(X,G,NDV)
      WRITE(*,10)
10  FORMAT(' NO.      COST FUNCT      STEP SIZE',
&        ' NORM OF GRAD  ')
      DO 20 K = 1, NOC
          CALL HASN(X,H,NDV)
          CALL SCALE (G,D,-1.0D0,NDV)
          CALL SYSEQ(H,NDV,D)
          IF (DOT(G,D,NDV) .GE. 1.0E-8) GO TO 60
          CALL GOLDM(X,D,WK,ALFA,DELTA,EPSLON,F,NCOUNT,NDV)
          CALL SCALE(D,D,ALFA,NDV)
          CALL PRINT(K,X,ALFA,G,F,NDV)

```

FIGURE D-4 A program for Newton's method.

```

        CALL ADD(X,D,X,NDV)
        CALL GRAD(X,G,NDV)
        IF(TNORM(G,NDV) .LE. EPSL) GO TO 30
20    CONTINUE

        WRITE(*,*)
        WRITE(*,*) ' LIMIT ON NO. OF CYCLES HAS EXCEEDED'
        WRITE(*,*) ' THE CURRENT DESIGN VARIABLES ARE:'
        WRITE(*,*) X
        CALL EXIT
30    WRITE(*,*)
        WRITE(*,*) 'THE OPTIMAL DESIGN VARIABLES ARE  : '
        WRITE(*,40) X
40    FORMAT(4X,3F15.6)
        CALL FUNCT(X,F,NCOUNT,NDV)
        WRITE(*,50) ' OPTIMUM COST FUNCTION VALUE IS      :', F
50    FORMAT(A, F13.6)
        WRITE(*,*) 'NO. OF FUNCTION EVALUATIONS ARE      :      ', NCOUNT
        CALL EXIT

60    WRITE(*,*)
        WRITE(*,*) ' DESCENT DIRECTION CANNOT BE FOUND'
        WRITE(*,*) ' THE CURRENT DESIGN VARIABLES ARE:'
        WRITE(*,40) X

        STOP
        END

        DOUBLE PRECISION FUNCTION DOT(X,Y,N)
C
C    CALCULATES DOT PRODUCT OF VECTORS X AND Y
C
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        DIMENSION X(N),Y(N)

        SUM = 0.0D0
        DO 10 I = 1, N
        SUM = SUM + X(I) * Y(I)
10    CONTINUE
        DOT = SUM

        RETURN
        END

```

FIGURE D-4 Continued

```

SUBROUTINE HASN(X,H,N)
C
C   CALCULATES THE HESSIAN MATRIX H AT X
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(N),H(N,N)

      H(1,1) = 2.0D0
      H(2,2) = 4.0D0
      H(3,3) = 4.0D0
      H(1,2) = 2.0D0
      H(1,3) = 0.0D0
      H(2,3) = 2.0D0
      H(2,1) = H(1,2)
      H(3,1) = H(1,3)
      H(3,2) = H(2,3)
      RETURN
      END

SUBROUTINE SYSEQ(A,N,B)
C
C   SOLVES AN N X N SYMMETRIC SYSTEM OF LINEAR EQUATIONS
C   AX = B
C   A IS THE COEFFICIENT MATRIX; B IS THE RIGHT HAND SIDE;
C   THESE ARE INPUT

C   B CONTAINS SOLUTION ON RETURN
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION A(N,N), B(N)

C   REDUCTION OF EQUATIONS
C
      M = 0
50    M = M + 1
      MM = M + 1
      B(M) = B(M) / A(M,M)
      IF (M - N) 70, 130, 70
70    DO 80 J = MM, N
          A(M,J) = A(M,J) / A(M,M)
80    CONTINUE
C
C   SUBSTITUTION INTO REMAINING EQUATIONS
C

```

FIGURE D-4 Continued

```

        DO 120 I = MM, N
          IF(A(I,M)) 90, 120, 90
90      DO 100 J = I, N
          A(I,J) = A(I,J) - A(I,M) * A(M,J)
          A(J,I) = A(I,J)
100     CONTINUE
          B(I) = B(I) - A(I,M) * B(M)
120     CONTINUE
        GO TO 50

C
C   BACK SUBSTITUTION
C
130    M = M - 1
        IF(M .EQ. 0) GO TO 150
        MM = M + 1
        DO 140 J = MM, N
          B(M) = B(M) - A(M,J) * B(J)
140    CONTINUE
        GO TO 130

150    RETURN
        END

```

FIGURE D-4 Continued

References

- AA (1986). *Construction manual series. Section 1, No. 30*. Washington, D.C.: The Aluminum Association.
- Abadie, J. (Ed.). (1970). *Nonlinear programming*. North Holland, Amsterdam.
- Abadie, J., & Carpenter, J. (1969). Generalization of the Wolfe reduced gradient method to the case of nonlinear constraints. In R. Fletcher (Ed.), *Optimization* (pp. 37–47). New York: Academic Press.
- Adelman, H., & Haftka, R. T. (1986). Sensitivity analysis of discrete structural systems. *AIAA Journal*, 24(5), 823–832.
- AISC (1989). *Manual of steel construction: Allowable stress design* (9th ed.). Chicago: American Institute of Steel Construction.
- Al-Saadoun, S. S., & Arora, J. S. (1989). Interactive design optimization of framed structures, *Journal of Computing in Civil Engineering, ASCE*, 3(1), 60–74.
- Arora, J. S. (1984). An algorithm for optimum structural design without line search. In E. Atrek, R. H. Gallagher, K. M. Ragsdell, & O. C. Zienkiewicz (Eds.), *New directions in optimum structural design* (pp. 429–441). New York: John Wiley & Sons.
- (1995). Structural design sensitivity analysis: Continuum and discrete approaches. In J. Herskovits (Ed.), *Advances in Structural Optimization* (pp. 47–70). Boston: Kluwer Academic Publishers.
- (1999). Optimization of structures subjected to dynamic loads. In C. T. Leondes (Ed.), *Structural dynamic systems: Computational techniques and optimization*, (Vol. 7, pp. 1–73). Newark, NJ: Gordon & Breech Publishers.
- (2002). Methods for discrete variable structural optimization. In S. Burns (Ed.), *Recent advances in optimal structural design* (pp. 1–40). Reston, VA: Structural Engineering Institute, ASCE.
- Arora, J. S. (Ed.). (1997). *Guide to structural optimization, ASCE Manuals and Reports on Engineering Practice, No. 90*. Reston, VA: American Society of Civil Engineering.
- Arora, J. S., & Haug, E. J. (1979). Methods of design sensitivity analysis in structural optimization. *AIAA Journal*, 17(9), 970–974.
- Arora, J. S., & Huang, M. W. (1996). Discrete structural optimization with commercially available sections: A review. *Journal of Structural and Earthquake Engineering, JSCE*, 13(2), 93–110.

- Arora, J. S., & Thanedar, P. B. (1986). Computational methods for optimum design of large complex systems. *Computational Mechanics*, 1(2), 221–242.
- Arora, J. S., & Tseng, C. H. (1987a). *User's manual for IDESIGN: Version 3.5*. Iowa City: Optimal Design Laboratory, College of Engineering, The University of Iowa.
- (1987b). An investigation of Pshenichnyi's recursive quadratic programming method for engineering optimization—A discussion. *Journal of Mechanisms, Transmissions and Automation in Design, Transactions of the ASME*, 109(6), 254–256.
- Arora, J. S., & Tseng, C. H. (1988). Interactive design optimization. *Engineering Optimization*, 13, 173–188.
- Arora, J. S., Burns, S., & Huang, M. W. (1997). What is optimization? In J. S. Arora (Ed.), *Guide to structural optimization, ASCE Manual on Engineering Practice, No. 90* (pp. 1–23). Reston, VA: American Society of Civil Engineers.
- Arora, J. S., Chahande, A. I., & Paeng, J. K. (1991). Multiplier methods for engineering optimization. *International Journal for Numerical Methods in Engineering*, 32, 1485–1525.
- Arora, J. S., Huang, M. W., & Hsieh, C. C. (1994). Methods for optimization of nonlinear problems with discrete variables: A review. *Structural Optimization*, 8(2/3), 69–85.
- Arora, J. S., Elwakeil, O. A., Chahande, A. I., & Hsieh, C. C. (1995). Global optimization methods for engineering applications: A review. *Structural Optimization*, 9, 137–159.
- Athan, T. W., & Papalambros, P. Y. (1996). A note on weighted criteria methods for compromise solutions in multi-objective optimization. *Engineering Optimization*, 27, 155–176.
- Atkinson, K. E. (1978). *An introduction to numerical analysis*. New York: John Wiley & Sons.
- Balling, R. J. (2000). Pareto sets in decision-based design. *Journal of Engineering Valuation and Cost Analysis*, 3(2), 189–198.
- (2003). The maximin fitness function: Multi-objective city and regional planning. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, & L. Thiele (Eds.), *Second International Conference on Evolutionary Multi-Criterion Optimization*, Faro, Portugal, April 8–11, 2003, Berlin: Springer Publishing, 1–15.
- Balling, R. J., Taber, J. T., Brown, M. R., & Day, K. (1999). Multiobjective urban planning using genetic algorithm. *Journal of Urban Planning and Development*, 125(2), 86–99.
- Balling, R. J., Taber, J. T., Day, K., & Wilson, S. (2000). Land use and transportation planning for twin cities using a genetic algorithm. *Transportation Research Record*, 1722, 67–74.
- Bartel, D. L. (1969). *Optimum design of spatial structures*. (Doctoral Dissertation, College of Engineering, The University of Iowa, Iowa City).
- Belegundu, A. D., & Arora, J. S. (1984a). A recursive quadratic programming algorithm with active set strategy for optimal design. *International Journal for Numerical Methods in Engineering*, 20(5), 803–816.
- (1984b). A computational study of transformation methods for optimal design. *AIAA Journal*, 22(4), 535–542.
- (1985). A study of mathematical programming methods for structural optimization. *International Journal for Numerical Methods in Engineering*, 21(9), 1583–1624.
- Bell, W. W. (1975). *Matrices for scientists and engineers*. New York: Van Nostrand Reinhold.
- Blank, L., & Tarquin, A. (1983). *Engineering economy* (2nd ed.). New York: McGraw-Hill.
- Branin, F. H., & Hoo, S. K. (1972). A method for finding multiple extrema of a function of n variables. In F. A. Lootsma (Ed.), *Numerical methods of nonlinear optimization*. London: Academic Press.
- Carmichael, D. G. (1980). Computation of pareto optima in structural design. *International Journal for Numerical Methods in Engineering*, 15, 925–952.

- Chandrupatla, T. R., & Belegundu, A. D. (1997). *Introduction to finite elements in engineering* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.
- Chen, S. Y., & Rajan, S. D. (2000). A robust genetic algorithm for structural optimization. *Structural Engineering and Mechanics*, 10, 313–336.
- Chen, W., Sahai, A., Messac, A., & Sundararaj, G. (2000). Exploration of the effectiveness of physical programming in robust design. *Journal of Mechanical Design*, 122, 155–163.
- Cheng, F. Y., & Li, D. (1997). Multiobjective optimization design with pareto genetic algorithm. *Journal of Structural Engineering*, 123, 1252–1261.
- (1998). Genetic algorithm development for multiobjective optimization of structures. *AIAA Journal*, 36, 1105–1112.
- Chopra, A. K. (1995). *Dynamics of structures: Theory and applications to earthquake engineering*, Upper Saddle River, NJ: Prentice-Hall.
- Clough, R. W., & Penzien, J. (1975). *Dynamics of structures*. New York: McGraw-Hill.
- Coello-Coello, C. A., Van Veldhuizen, D. A., & Lamont, G. B. (2002). *Evolutionary algorithms for solving multi-objective problems*. New York: Kluwer Academic Publishers.
- Cohon, J. L. (1978) *Multiobjective programming and planning*. New York: Academic Press.
- Cook, R. D. (1981). *Concepts and applications of finite element analysis*. New York: John Wiley & Sons.
- Cooper, L., & Steinberg, D. (1970). *Introduction to methods of optimization*. Philadelphia: W. B. Saunders.
- Corcoran, P. J. (1970). Configuration optimization of structures. *International Journal of Mechanical Sciences*, 12, 459–462.
- Crandall, S. H., Dahl, H. C., & Lardner, T. J. (1978). *Introduction to mechanics of solids*. New York: McGraw-Hill.
- Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. *Computer Journal*, 8, 250–255.
- Das, I., & Dennis, J. E. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14, 63–69.
- Davidon, W. C. (1959). *Variable metric method for minimization* (Research and Development Report ANL-5990). Argonne, Illinois: Argonne National Laboratory.
- De Boor, C. (1978). *A practical guide to splines: Applied mathematical sciences* (Vol. 27). New York: Springer-Verlag.
- Deb, K. (1989). Genetic algorithms in multimodal function optimization. *Masters Thesis*. Tuscaloosa, AL: University of Alabama (TCGA Report No. 89002).
- (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester, UK: John Wiley & Sons.
- Deif, A. S. (1982). *Advanced matrix theory for scientists and engineers*. New York: Halsted Press.
- Dixon, L. C. W., & Szego, G. P. (Eds.). (1978). *Towards global optimization 2*. North Holland Publishers, Amsterdam.
- Ehrgott, M., & Gandibleux, X. (Eds.). (2002). *Multiple criteria optimization: state of the art annotated bibliographic surveys*. Boston: Kluwer Academic Publishers.
- Elwakeil, O. A. (1995). *Algorithms for global optimization and their application to structural optimization problems*. (Doctoral Dissertation, The University of Iowa, Iowa City).
- Elwakeil, O. A., & Arora, J. S. (1995). Methods for finding feasible points in constrained optimization. *AIAA Journal*, 33(9), 1715–1719.
- (1996a). Two algorithms for global optimization of general NLP problems. *International Journal for Numerical Methods in Engineering*, 39, 3305–3325.

- (1996b). Global optimization of structural systems using two new methods. *Structural Optimization*, 12, 1–12.
- Evtushenko, Yu. G. (1974). Methods of search for the global extremum. *Operations Research, Computing Center of the U.S.S.R Akad. of Sci.*, 4, 39–68.
- (1985). *Numerical optimization techniques*. New York: Optimization Software.
- Fiacco, A. V., & McCormick, G. P. (1968). *Nonlinear programming: Sequential unconstrained minimization techniques*, Philadelphia: Society for Industrial and Applied Mathematics.
- Fletcher, R., & Powell, M. J. D. (1963). A rapidly convergent descent method for minimization. *The Computer Journal*, 6, 163–180.
- Fletcher, R., & Reeves, R. M. (1964). Function minimization by conjugate gradients. *The Computer Journal*, 7, 149–160.
- Floudas, C. A., et al. (1999). *Handbook of test problems in local and global optimization*. Norwell, MA: Kluwer Academic Publishers.
- Fonseca, C. M., & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization. *The Fifth International Conference on Genetic Algorithms (Urbana-Champaign, IL)*, San Mateo, CA: Morgan Kaufmann Publishers, 416–423.
- Forsythe, G. E., & Moler, C. B. (1967). *Computer solution of linear algebraic systems*. Englewood Cliffs, NJ: Prentice-Hall.
- Franklin, J. N. (1968). *Matrix theory*. Englewood Cliffs, NJ: Prentice-Hall.
- Gabrielle, G. A., & Beltracchi, T. J. (1987). An investigation of Pschenichnyi's recursive quadratic programming method for engineering optimization. *Journal of Mechanisms, Transmissions and Automation in Design, Transactions of the ASME*, 109(6), 248–253.
- Gen, M., & Cheng, R. (1997). *Genetic algorithms and engineering design*. New York: John Wiley & Sons.
- Gere, J. M., & Weaver, W. (1983). *Matrix algebra for engineers*. Monterey, CA: Brooks/Cole Engineering Division.
- Gill, P. E., Murray, W., & Wright, M. H. (1981). *Practical optimization*. New York: Academic Press.
- (1991). *Numerical linear algebra and optimization* (Vol. 1). New York: Addison-Wesley.
- Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison-Wesley.
- Grandin, H. (1986). *Fundamentals of the finite element method*. New York: Macmillan.
- Grant, E. L., Ireson, W. G., & Leavenworth, R. S. (1982). *Principles of engineering economy* (7th ed.). New York: John Wiley & Sons.
- Hadley, G. (1964). *Nonlinear and dynamic programming*. Reading, MA: Addison-Wesley.
- Han, S. P. (1976). Superlinearly convergent variable metric algorithms for general nonlinear programming. *Mathematical Programming*, 11, 263–282.
- (1977). A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications*, 22, 297–309.
- Haug, E. J., & Arora, J. S. (1979). *Applied optimal design*. New York: Wiley Interscience.
- Hock, W., & Schittkowski, K. (1981). *Test examples for nonlinear programming codes*, (Lecture Notes in Economics and Mathematical Systems, 187). New York: Springer-Verlag.
- Hohn, F. E. (1964). *Elementary matrix algebra*. New York: Macmillan.
- Holland, J. H. (1975). *Adaptation in natural and artificial system*. Ann Arbor, MI: The University of Michigan Press.
- Hopper, M. J. (1981). *Harwell subroutine library*. Oxfordshire, UK: Computer Science and Systems Division, AERE Harwell.

- Horn, J., Nafpliotis, N., & Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. *The First IEEE Conference on Evolutionary Computation (Orlando, FL)*. Piscataway, NJ: IEEE Neural Networks Council, 82–87.
- Hsieh, C. C., & Arora, J. S. (1984). Design sensitivity analysis and optimization of dynamic response. *Computer Methods in Applied Mechanics and Engineering*, 43, 195–219.
- Huang, M. W., & Arora, J. S. (1995). Engineering optimization with discrete variables. *Proceedings of the 36th AIAA SDM Conference*, New Orleans, April 10–12, 1475–1485.
- (1996). A self-scaling implicit SQP method for large scale structural optimization. *International Journal for Numerical Methods in Engineering*, 39, 1933–1953.
- (1997a). Optimal design with discrete variables: Some numerical experiments. *International Journal for Numerical Methods in Engineering*, 40, 165–188.
- (1997b). Optimal design of steel structures using standard sections. *Structural and Multidisciplinary Optimization*, 14, 24–35.
- Huang, M. W., Hsieh, C. C., & Arora, J. S. (1997). A genetic algorithm for sequencing type problems in engineering design. *International Journal for Numerical Methods in Engineering*, 40, 3105–3115.
- Huebner, K. H., & Thornton, E. A. (1982). *The finite element method for engineers*. New York: John Wiley & Sons.
- Ishibuchi, H., & Murata, T. (1996). Multiobjective genetic local search algorithm. *1996 IEEE International Conference on Evolutionary Computation (Nagoya, Japan)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, 119–124.
- Iyengar, N. G. R., & Gupta, S. K. (1980). *Programming methods in structural design*. New York: John Wiley & Sons.
- Javonovic, V., & Kazerounian, K. (2000). Optimal design using chaotic descent method. *Journal of Mechanical Design, ASME*, 122(3), 137–152.
- Karush, W. (1939). Minima of functions of several variables with inequalities as side constraints. *Masters Thesis*. Chicago: Department of Mathematics, University of Chicago.
- Kim, C. H., and Arora, J. S. (2003). Development of simplified dynamic models using optimization: Application to crushed tubes. *Computer Methods in Applied Mechanics and Engineering*, 192(16–18), 2073–2097.
- Kocer, F. Y., & Arora, J. S. (1996a). Design of prestressed concrete poles: An optimization approach. *Journal of Structural Engineering, ASCE*, 122(7), 804–814.
- (1996b). Optimal design of steel transmission poles. *Journal of Structural Engineering, ASCE*, 122(11), 1347–1356.
- (1997). Standardization of transmission pole design using discrete optimization methods. *Journal of Structural Engineering, ASCE*, 123(3), 345–349.
- (1999). Optimal design of H-frame transmission poles subjected to earthquake loading. *Journal of Structural Engineering, ASCE*, 125(11) 1299–1308.
- (2002). Optimal design of latticed towers subjected to earthquake loading. *Journal of Structural Engineering, ASCE*, 128(2), 197–204.
- Koski, J. (1985). Defectiveness of weighting method in multicriterion optimization of structures. *Communications in Applied Numerical Methods*, 1, 333–337.
- Kunzi, H. P., & Krelle, W. (1966). *Nonlinear programming*. Waltham, MA: Blaisdell Publishing.
- Land, A. M., & Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28, 497–520.
- Lee, S. M., & Olson, D. L. (1999). Goal programming. In T. Gal, T. J. Stewart, & T. Hanne (Eds.), *Multicriteria decision making: Advances in MCDM models, algorithms, theory, and applications*. Boston: Kluwer Academic Publishers.
- Lemke, C. E. (1965). Bimatrix equilibrium points and mathematical programming. *Management Science*, 11, 681–689.

- Levy, A. V., & Gomez, S. (1985). The tunneling method applied to global optimization. In P. T. Boggs, R. H. Byrd, & R. B. Schnabel (Eds.), *Numerical optimization 1984*, Philadelphia: Society for Industrial and Applied Mathematics.
- Lim, O. K., & Arora, J. S. (1986). An active set RQP algorithm for optimal design. *Computer Methods in Applied Mechanics and Engineering*, 57, 51–65.
- (1987). Dynamic response optimization using an active set RQP algorithm. *International Journal for Numerical Methods in Engineering*, 24(10), 1827–1840.
- Lucidi, S., & Piccioni, M. (1989). Random tunneling by means of acceptance-rejection sampling for global optimization. *Journal of Optimization Theory and Applications*, 62(2), 255–277.
- Luenberger, D. G. (1984). *Linear and nonlinear programming*. Reading, MA: Addison-Wesley.
- Marler, T. R., & Arora, J. S. (in press). Survey of multiobjective optimization methods for engineering. *Structural and Multidisciplinary Optimization*.
- Marquardt, D. W. (1963). An algorithm for least squares estimation of nonlinear parameters. *SIAM Journal*, 11, 431–441.
- McCormick, G. P. (1967). Second order conditions for constrained optima. *SIAM Journal Applied Mathematics*, 15, 641–652.
- Meirovitch, L. (1985). *Introduction to dynamics and controls*. New York: John Wiley & Sons.
- Messac, A. (1996). Physical programming: Effective optimization for computational design. *AIAA Journal*, 34(1), 149–158.
- Messac, A., & Mattson, C. A. (2002). Generating well-distributed sets of pareto points for engineering design using physical programming. *Optimization and Engineering*, 3, 431–450.
- Messac, A., Puemi-Sukam, C., & Melachrinoudis, E. (2000a). Aggregate objective functions and pareto frontiers: Required relationships and practical implications. *Optimization and Engineering*, 1, 171–188.
- Messac, A., Puemi-Sukam, C., & Melachrinoudis, E. (2001). Mathematical and pragmatic perspectives of physical programming. *AIAA Journal*, 39(5), 885–893.
- Messac, A., Sundararaj, G. J., Tappeta, R. V., & Renaud, J. E. (2000b). Ability of objective functions to generate points on nonconvex pareto frontiers. *AIAA Journal*, 38(6), 1084–1091.
- Mitchell, M. (1996). *An introduction to genetic algorithms*. Cambridge, MA: MIT Press.
- Murata, T., Ishibuchi, H., & Tanaka, H. (1996). Multiobjective genetic algorithm and its applications to flowshop scheduling. *Computers and Industrial Engineering*, 30, 957–968.
- NAG (1984). *FORTTRAN library manual*. Downers Grove, IL: Numerical Algorithms Group.
- Narayana, S., & Azarm, S. (1999). On improving multiobjective genetic algorithms for design optimization. *Structural Optimization*, 18, 146–155.
- Nelder, J. A., & Mead, R. A. (1965). A simplex method for function minimization. *Computer Journal*, 7, 308–313.
- Nocedal, J., & Wright, S. J. (2000). *Numerical optimization*. New York: Springer-Verlag.
- Norton, R. L. (2000). *Machine design: An integrated approach* (2nd ed.). Upper Saddle River, NJ: Prentice-Hall.
- Osman, M. O. M., Sankar, S., & Dukkipati, R. V. (1978). Design synthesis of a multi-speed machine tool gear transmission using multiparameter optimization. *Journal of Mechanical Design, Transactions of ASME*, 100, 303–310.
- Oszyczka, A. (2002). *Evolutionary algorithms for single and multicriteria design optimization*. Berlin, Germany: Physica Verlag.
- Pardalos, P. M., & Rosen J. B. (1987). *Constrained global optimization: Algorithms and applications*. In G. Goos & J. Hartmanis (Eds.), *Lecture Notes in Computer Science*. New York: Springer-Verlag.

- Pardalos, P. M., Migdalas, A., & Burkard, R. (2002). *Combinatorial and global optimization, Series on Applied Mathematics* (Vol. 14). River Edge, NJ: World Scientific Publishing.
- Pardalos, P. M., Romeijn, H. E., & Tuy, H. (2000). Recent developments and trends in global optimization. *Journal of Computational and Applied Mathematics*, 124, 209–228.
- Pareto, V. (1971) *Manuale di economica politica, societa editrice libraria* (A. S. Schwier, A. N., Page, & A. M. Kelley, Eds., Trans.). New York: Augustus M. Kelley Publishers (Original work published 1906).
- Pederson, D. R., Brand, R. A., Cheng, C., & Arora, J. S. (1987). Direct comparison of muscle force predictions using linear and nonlinear programming. *Journal of Biomechanical Engineering, Transactions of the ASME*, 109(3), 192–199.
- Pezeshek, S., & Camp, C. V. (2002). State-of-the-art on use of genetic algorithms in design of steel structures. In S. Burns (Ed.), *Recent advances in optimal structural design*. Reston, VA: Structural Engineering Institute, ASCE.
- Price, W. L. (1987). Global optimization algorithms for a CAD workstation. *Journal of Optimization Theory and Applications*, 55, 133–146.
- Pshenichny, B. N., & Danilin, Y. M. (1982). *Numerical methods in extremal problems* (2nd ed.). Moscow: Mir Publishers.
- Ravindran, A., & Lee, H. (1981). Computer experiments on quadratic programming algorithms. *European Journal of Operations Research*, 8(2), 166–174.
- Reklaitis, G. V., Ravindran, A., & Ragsdell, K. M. (1983). *Engineering optimization: Methods and applications*. New York: John Wiley & Sons.
- Rinnooy, A. H. G., & Timmer, G. T. (1987a). Stochastic global optimization methods. Part I: Clustering methods. *Mathematical Programming*, 39, 27–56.
- (1987b). Stochastic global optimization methods. Part II: Multilevel methods. *Mathematical Programming*, 39, 57–78.
- Sargeant, R. W. H. (1974). Reduced-gradient and projection methods for nonlinear programming. In P. E. Gill, & W. Murray (Eds.), *Numerical methods for constrained optimization* (pp. 149–174). New York: Academic Press.
- Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated GENETIC algorithms. *The First International Conference on Genetic Algorithms and Their Applications (Pittsburgh, PA)*. Hillsdale, NJ: Lawrence Erlbaum Associates, 93–100.
- Schittkowski, K. (1981). The nonlinear programming method of Wilson, Han and Powell with an augmented Lagrangian type line search function, Part 1: Convergence analysis, Part 2: An efficient implementation with linear least squares subproblems. *Numerische Mathematik*, 38, 83–127.
- (1987). *More test examples for nonlinear programming codes*. New York: Springer-Verlag.
- Schmit, L. A. (1960). Structural design by systematic synthesis. *Proceedings of the Second ASCE Conference on Electronic Computations (Pittsburgh, PA)*. Reston, VA: American Society of Civil Engineers, 105–122.
- Schrage, L. (1981). *User's manual for LINDO*. Palo Alto, CA: The Scientific Press.
- Schrage, L. (1991). *LINDO: Text and software*. Palo Alto, CA: The Scientific Press.
- Schrijver, A. (1986). *Theory of linear and integer programming*. New York: John Wiley & Sons.
- Shigley, J. E. (1977). *Mechanical engineering design*. New York: McGraw-Hill.
- Shigley, J. E., & Mischke, C. R. (2001). *Mechanical engineering design* (6th ed.). New York: McGraw-Hill.
- Siddall, J. N. (1972). *Analytical decision-making in engineering design*. Englewood Cliffs, NJ: Prentice Hall.
- Spotts, M. F. (1953). *Design of machine elements* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.

- Srinivas, N., & Deb, K. (1995). Multiobjective optimization using nondominated sorting in general algorithms, *Evolutionary Computations*, 2, 221–248.
- Stadler, W. (1977). Natural structural shapes of shallow arches. *Journal of Applied Mechanics*, 44, 291–298.
- (1988). Fundamentals of multicriteria optimization. In W. Stadler (Ed.), *Multicriteria optimization in engineering and in the sciences* (pp. 1–25). New York: Plenum Press.
- (1995). Caveats and boons of multicriteria optimization. *Microcomputers in Civil Engineering*, 10, 291–299.
- Stadler, W., & Dauer, J. P. (1992). Multicriteria optimization in engineering: A tutorial and Survey. In M. P. Kamat (Ed.), *Structural optimization: Status and promise* (pp. 211–249). Washington, D.C.: American Institute of Aeronautics and Astronautics.
- Stewart, G. (1973). *Introduction to matrix computations*. New York: Academic Press.
- Strang, G. (1976). *Linear algebra and its applications*. New York: Academic Press.
- Syslo, M. M., Deo, N., & Kowalik, J. S. (1983). *Discrete optimization algorithms*. Englewood Cliffs, NJ: Prentice-Hall.
- Thanedar, P. B., Arora, J. S., & Tseng, C. H. (1986). A hybrid optimization method and its role in computer aided design. *Computers and Structures*, 23(3), 305–314.
- Thanedar, P. B., Arora, J. S., Tseng, C. H., Lim, O. K., & Park, G. J. (1987). Performance of some SQP algorithms on structural design problems. *International Journal for Numerical Methods in Engineering*, 23(12), 2187–2203.
- Törn, A., & Žilinskas, A. (1989). *Global optimization*. In G. Goos, & J. Hartmanis (Eds.), *Lecture Notes in Computer Science*. New York: Springer-Verlag.
- Tseng, C. H., & Arora, J. S. (1987). *Optimal design for dynamics and control using a sequential quadratic programming algorithm*. Technical report No. ODL-87.10. Iowa City: Optimal Design Laboratory, College of Engineering, The University of Iowa.
- (1988). On implementation of computational algorithms for optimal design 1: Preliminary investigation; 2: Extensive numerical investigation. *International Journal for Numerical Methods in Engineering*, 26(6), 1365–1402.
- Wahl, A. M. (1963). *Mechanical springs* (2nd ed.). New York: McGraw-Hill.
- Walster, G. W., Hansen, E. R., & Sengupta, S. (1984). Test results for a global optimization algorithm. In T. Boggs, et al. (Eds.), *Numerical optimization* (pp. 280–283). Philadelphia: SIAM.
- Wilson, R. B. (1963). *A simplicial algorithm for concave programming*. (Doctoral Dissertation, Graduate School of Business Administration, Harvard University, Boston, MA).
- Wolfe, P. (1959). The simplex method for quadratic programming. *Econometrica*, 27(3), 382–398.
- Zhou, C. S., & Chen, T. L. (1997). Chaotic annealing and optimization. *Physical Review E*, 55(3), 2580–2587.

Bibliography

- AASHTO (1992). *Standard specifications for highway bridges* (15th ed.). Washington, D.C.: American Association of State Highway and Transportation Officials.
- Ackoff, R. L., & Sasieni, M. W. (1968). *Fundamentals of operations research*. New York: John Wiley & Sons.
- Aoki, M. (1971). *Introduction to optimization techniques*. New York: Macmillan.
- Arora, J. S. (1990a). Computational design optimization: A review and future directions. *Structural Safety*, 7, 131–148.
- (1990b). Global optimization methods for engineering design. *Proceedings of the 31st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Long Beach, CA*, 123–135.
- Arora, J. S., & Baenziger, G. (1986). Uses of artificial intelligence in design optimization. *Computer Methods in Applied Mechanics and Engineering*, 54, 303–323.
- (1987). A nonlinear optimization expert system. In D. R. Jenkins (Ed.), *Proceedings of the ASCE Structures Congress' 87, Computer Applications in Structural Engineering*, 113–125.
- ASTM (1980). *Standard metric practice, No. E380-79*, Philadelphia: American Society for Testing and Material.
- Belegundu, A. D., & Chandrupatla, T. R. (1999). *Optimization concepts and applications in engineering*. Upper Saddle River, NJ: Prentice Hall.
- Bertsekas, D. P. (1995). *Nonlinear programming*. Belmont, MA: Athena Scientific.
- Bhatti, M. A. (2000). *Practical optimization methods with Mathematica applications*. New York: Springer Telos.
- Cauchy, A. (1847). Method generale pour la resolution des systemes d'equations simultanees. *Comptes Rendus. de Academie Scientifique*, 25, 536–538.
- Chong, K. P., & Zak, S. H. (2001). *An introduction to optimization* (2nd ed.). New York: John Wiley & Sons.
- Dano, S. (1974). *Linear programming in industry* (4th ed.). New York: Springer-Verlag.
- Dantzig, G. B., & Thapa, M. N. (1997). *Linear programming, 1: Introduction*. New York: Springer-Verlag.
- Day, H. J., & Dolbear, F. (1965). Regional water quality management. *Proceedings of the 1st Annual Meeting of the American Water Resources Association*, Chicago: University of Chicago, 283–309.

- Deiningner, R. A. (1975). Water quality management—The planning of economically optimal pollution control systems. *Proceedings of the 1st Annual Meeting of the American Water Resources Association*, Chicago: University of Chicago, 254–282.
- Drew, D. (1968). *Traffic flow theory and control*. New York: McGraw-Hill.
- Fang, S. C., & Puthenpura, S. (1993). *Linear optimization and extensions: Theory and algorithms*. Englewood Cliffs, NJ: Prentice-Hall.
- Gill, P. E., Murray, W., Saunders, M. A., & Wright, M. H. (1984). *User's guide for QPSOL: Version 3.2*. Stanford, CA: Systems Optimization Laboratory, Department of Operations Research, Stanford University.
- Hadley, G. (1961). *Linear programming*. Reading, MA: Addison-Wesley.
- Hafika, R. T., & Gurdal, Z. (1992). *Elements of structural optimization*. Norwell, MA: Kluwer Academic Publishers.
- Hafika, R. T., & Kamat, M. P. (1985). *Elements of structural optimization*. Dordrecht, Holland: Martinus Nijhoff.
- Hock, W., & Schittkowski, K. (1983). A comparative performance evaluation of 27 nonlinear programming codes. *Computing*, 30, 335–358.
- Hyman, B. (2003). *Fundamental of engineering design* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.
- Jennings, A. (1977). *Matrix computations for engineers*. New York: John Wiley & Sons.
- Kirsch, U. (1981). *Optimum structural design*. New York: McGraw-Hill.
- (1993). *Structural optimization*. New York: Springer-Verlag.
- Lynn, W. R. (1964). State development of wastewater treatment works. *Journal of Water Pollution Control Federation*, 722–751.
- MathWorks (2001). *Optimization toolbox for use with MATLAB, User's Guide, Ver. 2*. Natick, MA: The MathWorks, Inc.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21, 1087–1092.
- Microsoft EXCEL, Version 11.0, Redmond, WA: Microsoft Corporation.
- Minoux, M. (1986). *Mathematical programming theory and algorithms*. New York: John Wiley & Sons.
- Moré J. J., & Wright, S. J. (1993). *Optimization software guide*. Philadelphia: Society for Industrial and Applied Mathematics.
- Nash, S. G., & Sofer, A. (1996). *Linear and nonlinear programming*. New York: McGraw-Hill.
- Nemhauser, G. L., & Wolsey, S. J. (1988). *Integer and combinatorial optimization*. New York: John Wiley & Sons.
- Onwubiko, C. (2000). *Introduction to engineering design optimization*. Upper Saddle River, NJ: Prentice Hall.
- Papalambros, P. Y., & Wilde, D. J. (2000). *Principles of optimal design: Modeling and computation* (2nd ed.). New York: Cambridge University Press.
- Powell, M. J. D. (1978a). A fast algorithm for nonlinearly constrained optimization calculations. In G. A. Watson, et al. (eds.), *Lecture Notes in Mathematics*. Berlin: Springer-Verlag. (Also in *Numerical Analysis*, Proceedings of the Biennial Conference held at Dundee, June 1977.)
- (1978b). The convergence of variable metric methods for nonlinearity constrained optimization calculations. In O. L. Mangasarian, R. R. Meyer, & S. M. Robinson (Eds.), *Nonlinear Programming 3*. New York: Academic Press.
- (1978c). Algorithms for nonlinear functions that use Lagrange functions. *Mathematical Programming*, 14, 224–248.

- Pshenichny, B. N. (1978). Algorithms for the general problem of mathematical programming. *Kibernetika*, 5, 120–125.
- Randolph, P. H., & Meeks, H. D. (1978). *Applied linear optimization*. Columbus, OH: GRID, Inc.
- Roark, R. J., & Young, W. C. (1975). *Formulas for stress and strain* (5th ed.). New York: McGraw-Hill.
- Rosen, J. B. (1961). The gradient projection method for nonlinear programming. *Journal of the Society for Industrial and Applied Mathematics*, 9, 514–532.
- Rubinstein, M. F., & Karagozian, J. (1966). Building design under linear programming. *Proceedings ASCE*, 92(ST6), 223–245.
- Salkin, H. M. (1975). *Integer programming*. Reading, MA: Addison-Wesley.
- Sasieni, M., Yaspan, A., & Friedman, L. (1960). *Operations-methods and problems*, New York: John Wiley & Sons.
- Shampine, L. F., & Gordon, M. K. (1975). *Computer simulation of ordinary differential equations: The initial value problem*. San Francisco: W. H. Freeman.
- Stark, R. M., & Nicholls, R. L. (1972). *Mathematical foundations for design: Civil engineering systems*. New York: McGraw-Hill.
- Stoecker, W. F. (1971). *Design of thermal systems*. New York: McGraw-Hill.
- Sun, P. F., Arora, J. S., & Haug, E. J. (1975). *Fail-safe optimal design of structures. Technical Report No. 19*. Iowa City: Department of Civil and Environmental Engineering, The University of Iowa.
- Vanderplaats, G. N. (1984). *Numerical optimization techniques for engineering design with applications*. New York: McGraw-Hill.
- Vanderplaats, G. N., & Yoshida, N. (1985). Efficient calculation of optimum design sensitivity. *AIAA Journal*, 23(11), 1798–1803.
- Venkataraman, P. (2002). *Applied optimization with MATLAB programming*. New York: John Wiley & Sons.
- Wohl, M., & Martin, B. V. (1967). *Traffic systems analysis*. New York: McGraw-Hill.
- Wu, N., & Coppins, R. (1981). *Linear programming and extensions*. New York: McGraw-Hill.
- Zoutendijk, G. (1960). *Methods of feasible directions*. Amsterdam: Elsevier.

Answers to Selected Problems

Chapter 3 Graphical Optimization

3.1 $\mathbf{x}^* = (2, 2)$, $f^* = 2$. **3.2** $\mathbf{x}^* = (0, 4)$, $F^* = 8$. **3.3** $\mathbf{x}^* = (8, 10)$, $f^* = 38$. **3.4** $\mathbf{x}^* = (4, 3.333, 2)$, $F^* = 11.33$. **3.5** $\mathbf{x}^* = (10, 10)$, $F^* = 400$. **3.6** $\mathbf{x}^* = (0, 0)$, $f^* = 0$. **3.7** $\mathbf{x}^* = (0, 0)$, $f^* = 0$. **3.8** $\mathbf{x}^* = (2, 3)$, $f^* = -22$. **3.9** $\mathbf{x}^* = (-2.5, 1.58)$, $f^* = -3.95$. **3.10** $\mathbf{x}^* = (-0.5, 0.167)$, $f^* = -0.5$. **3.21** $b^* = 24.66$ cm, $d^* = 49.32$ cm, $f^* = 1216$ cm³. **3.22** $R_o^* = 20$ cm, $R_i^* = 19.84$ cm, $f^* = 79.1$ kg. **3.23** $R^* = 53.6$ mm, $t^* = 5.0$ mm, $f^* = 66$ kg. **3.24** $R_o^* = 56$ mm, $R_i^* = 51$ mm, $f^* = 66$ kg. **3.25** $w^* = 93$ mm, $t^* = 5$ mm, $f^* = 70$ kg. **3.26** Infinite optimum points, $f^* = 0.812$ kg. **3.27** $A^* = 5000$, $h^* = 14$, $f^* = \$13.4$ million. **3.28** $R^* \cong 1.0$ m, $t^* = 0.0167$ m, $f^* \cong 8070$ kg. **3.29** $A_1^* = 6.1$ cm², $A_2^* = 2.0$ cm², $f^* = 5.39$ kg. **3.31** $t^* = 8.45$, $f^* = 1.91 \times 10^5$. **3.32** $R^* = 7.8$ m, $H^* = 15.6$ m, $f^* = \$1.75 \times 10^6$. **3.33** Infinite optimum points; one point: $R^* = 0.4$ m, $t^* = 1.59 \times 10^{-3}$ m, $f^* = 15.7$ kg. **3.34** For $l = 0.5$ m, $T_o = 10$ kN·m, $T_{\max} = 20$ kN·m, $x_1^* = 103$ mm, $x_2^* = 0.955$, $f^* = 2.9$ kg. **3.35** For $l = 0.5$, $T_o = 10$ kN·m, $T_{\max} = 20$ kN·m, $d_o^* = 103$ mm, $d_i^* = 98.36$ mm, $f^* = 2.9$ kg. **3.36** $R^* = 50.3$ mm, $t^* = 2.35$ mm, $f^* = 2.9$ kg. **3.37** $R^* = 20$ cm, $H^* = 7.2$ cm, $f^* = -9000$ cm³. **3.38** $R^* = 0.5$ cm, $N^* = 2550$, $f^* = -8000$ ($l = 10$). **3.39** $R^* = 33.7$ mm, $t^* = 5.0$ mm, $f^* = 41$ kg. **3.40** $R^* = 21.5$ mm, $t^* = 5.0$ mm, $f^* = 26$ kg. **3.41** $R^* = 27$, $t^* = 5$ mm, $f^* = 33$ kg. **3.42** $R_o^* = 36$ mm, $R_i^* = 31$ mm, $f^* = 41$ kg. **3.43** $R_o^* = 24.0$ mm, $R_i^* = 19.0$ mm, $f^* = 26$ kg. **3.44** $R_o^* = 29.5$ mm, $R_i^* = 24.5$ mm, $f^* = 33$ kg. **3.45** $D^* = 8.0$ cm, $H^* = 8.0$ cm, $f^* = 301.6$ cm². **3.46** $A_1^* = 413.68$ mm, $A_2^* = 163.7$ mm, $f^* = 5.7$ kg. **3.47** Infinite optimum points; one point: $R^* = 20$ mm, $t^* = 3.3$ mm, $f^* = 8.1$ kg. **3.48** $A^* = 390$ mm², $h^* = 500$ mm, $f^* = 5.5$ kg. **3.49** $A^* = 410$ mm², $s^* = 1500$ mm, $f^* = 8$ kg. **3.50** $A_1^* = 300$ mm², $A_2^* = 50$ mm², $f^* = 7$ kg. **3.51** $R^* = 130$ cm, $t^* = 2.86$ cm, $f^* = 57,000$ kg. **3.52** $d_o^* = 41.56$ cm, $d_i^* = 40.19$ cm, $f^* = 680$ kg. **3.53** $d_o^* = 1310$ mm, $t^* = 14.2$ mm, $f^* = 92,500$ N. **3.54** $H^* = 50.0$ cm, $D^* = 3.42$ cm, $f^* = 6.6$ kg.

Chapter 4 Optimum Design Concepts

4.2 $\cos x = 1.044 - 0.15175x - 0.35355x^2$ at $x = \pi/4$. **4.3** $\cos x = 1.1327 - 0.34243x - 0.25x^2$ at $x = \pi/3$. **4.4** $\sin x = -0.02199 + 1.12783x - 0.25x^2$ at $x = \frac{\pi}{6}$. **4.5** $\sin x = 0.06634 + 1.2625x - 0.35355x^2$ at $x = \frac{\pi}{4}$. **4.6** $e^x = 1 + x + 0.5x^2$ at $x = 0$. **4.7** $e^x = 7.389 - 7.389x + 3.6945x^2$ at $x = 2$. **4.8** $\bar{f}(x) = 41x_1^2 - 42x_1 - 40x_1x_2 + 20x_2 + 10x_2^2 + 15$; $\bar{f}(1.2, 0.8) = 7.64$, $f(1.2, 0.8) = 8.136$, Error = $f - \bar{f} = 0.496$. **4.9** Indefinite. **4.10** Indefinite. **4.11** Indefinite. **4.12** Positive definite. **4.13** Indefinite. **4.14** Indefinite. **4.15** Positive definite.

4.16 Indefinite. **4.22** $\mathbf{x} = (0, 0)$ – local minimum, $f = 7$. **4.23** $\mathbf{x} = (0, 0)$ – inflection point.
4.24 $\mathbf{x}^* = (-3.332, 0.0395)$ – local maximum, $f = 18.58$; $\mathbf{x}^* = (-0.398, 0.5404)$ – inflection point. **4.25** $\mathbf{x}^* = (4, 8)$ – inflection point; $\mathbf{x}^* = (-4, -8)$ – inflection point. **4.26** $x^* = (2n + 1)\pi$, $n = 0, \pm 1, \pm 2, \dots$ local minima, $f^* = -1$; $x^* = 2n\pi$, $n = 0, \pm 1, \pm 2, \dots$ local maxima, $f^* = 1$. **4.27** $\mathbf{x}^* = (0, 0)$ – local minimum, $f^* = 0$. **4.28** $x^* = 0$ – local minimum, $f^* = 0$; $x^* = 2$ – local maximum, $f^* = 0.541$. **4.29** $\mathbf{x}^* = (3.684, 0.7368)$ – local minimum, $f^* = 11.0521$. **4.30** $\mathbf{x}^* = (1, 1)$ – local minimum, $f^* = 1$. **4.31** $\mathbf{x}^* = (-\frac{2}{7}, -\frac{6}{7})$ – local minimum, $f^* = -\frac{24}{7}$. **4.32** $\mathbf{x}^* = (241.7643, 0.03099542)$ – local minimum, $U^* = 483,528.6$; $\mathbf{x}^* = (-241.7643, -0.03099542)$ – local maximum. **4.44** $\mathbf{x}^* = (2.166667, 1.833333)$, $\nu^* = -0.166667$, $f^* = -8.333333$. **4.49** $\mathbf{x}^* = (1.5088, 3.272)$, $\nu^* = -17.1503$, $f^* = 244.528$; $\mathbf{x}^* = (2.5945, -2.0198)$, $\nu^* = -1.4390$, $f^* = 15.291$; $\mathbf{x}^* = (-3.630, -3.1754)$, $\nu^* = -23.2885$, $f^* = 453.154$; $\mathbf{x}^* = (-3.7322, 3.0879)$, $\nu^* = -2.122$, $f^* = 37.877$. **4.50** $\mathbf{x}^* = (2, 2)$, $\nu^* = -2$, $f^* = 2$. **4.51** (i) No, (ii) Solution of equalities, $\mathbf{x}^* = (3, 1)$, $f^* = 4$. **4.53** $\mathbf{x}^* = (\frac{11}{6}, \frac{13}{6})$, $\nu^* = -\frac{23}{6}$, $F^* = -\frac{13}{6}$. **4.61** $\mathbf{x}^* = (0.816, 0.75)$, $\mathbf{u}^* = (0, 0, 0, 0)$, $f^* = 2.214$; $\mathbf{x}^* = (0.816, 0)$, $\mathbf{u}^* = (0, 0, 0, 3)$, $f^* = 1.0887$; $\mathbf{x}^* = (0, 0.75)$, $\mathbf{u}^* = (0, 0, 2, 0)$, $f^* = 1.125$; $\mathbf{x}^* = (1.5073, 1.2317)$, $\mathbf{u}^* = (0, 0.9632, 0, 0)$, $f^* = 0.251$; $\mathbf{x}^* = (1.0339, 1.655)$, $\mathbf{u}^* = (1.2067, 0, 0, 0)$, $f^* = 0.4496$; $\mathbf{x}^* = (0, 0)$, $\mathbf{u}^* = (0, 0, 2, 3)$, $f^* = 0$; $\mathbf{x}^* = (2, 0)$, $\mathbf{u}^* = (0, 2, 0, 7)$, $f^* = -4$; $\mathbf{x}^* = (0, 2)$, $\mathbf{u}^* = (\frac{5}{3}, 0, \frac{11}{3}, 0)$, $f^* = -2$; $\mathbf{x}^* = (1.386, 1.538)$, $\mathbf{u}^* = (0.633, 0.626, 0, 0)$, $f^* = -0.007388$. **4.62** $\mathbf{x}^* = (\frac{48}{23}, \frac{40}{23})$, $\mathbf{u}^* = 0$, $f^* = -\frac{192}{23}$. **4.63** $\mathbf{x}^* = (2.5, 1.5)$, $\mathbf{u}^* = 1$, $f^* = 1.5$. **4.64** $\mathbf{x}^* = (6.3, 1.733)$, $\mathbf{u}^* = (0, 0.8, 0, 0)$, $f^* = -56.901$. **4.65** $\mathbf{x}^* = (1, 1)$, $\mathbf{u}^* = 0$, $f^* = 0$. **4.66** $\mathbf{x}^* = (1, 1)$, $\mathbf{u}^* = (0, 0)$, $f^* = 0$. **4.67** $\mathbf{x}^* = (2, 1)$, $\mathbf{u}^* = (0, 2)$, $f^* = 1$. **4.68** $\mathbf{x}^* = (2.5945, 2.0198)$, $u_1^* = 1.439$, $f^* = 15.291$; $\mathbf{x}^* = (-3.63, 3.1754)$, $u_1^* = 23.2885$, $f^* = 453.154$; $\mathbf{x}^* = (1.5088, -3.2720)$, $u_1^* = 17.1503$, $f^* = 244.53$; $\mathbf{x}^* = (-3.7322, -3.0879)$, $u_1^* = 2.1222$, $f^* = 37.877$. **4.69** $\mathbf{x}^* = (3.25, 0.75)$, $\nu^* = -1.25$, $u^* = 0.75$, $f^* = 5.125$. **4.70** $\mathbf{x}^* = (\frac{4}{\sqrt{3}}, \frac{1}{3})$, $u^* = 0$, $f^* = -24.3$; $\mathbf{x}^* = (-\frac{4}{\sqrt{3}}, \frac{1}{3})$, $u^* = 0$, $f^* = 24.967$; $\mathbf{x}^* = (0, 3)$, $u^* = 16$, $f^* = -21$; $\mathbf{x}^* = (2, 1)$, $u^* = 4$, $f^* = -25$. **4.71** $\mathbf{x}^* = (-\frac{2}{7}, -\frac{6}{7})$, $u^* = 0$, $f^* = -\frac{24}{7}$. **4.72** $x^* = 4$, $y^* = 6$, $\mathbf{u}^* = (0, 0, 0, 0)$, $f^* = 0$. **4.74** Three local maxima: $x^* = 0$, $y^* = 0$, $\mathbf{u}^* = (0, 0, -18, -12)$, $F^* = 52$; $x^* = 6$, $y^* = 0$, $\mathbf{u}^* = (0, -4, 0, -12)$, $F^* = 40$; $x^* = 0$, $y^* = 12$, $\mathbf{u}^* = (-12, 0, -4, 0)$, $F^* = 52$; One stationary point: $x^* = 5$, $y^* = 7$, $\mathbf{u}^* = (-2, 0, 0, 0)$, $F^* = 2$. **4.79** $D^* = 7.98$ cm, $H^* = 8$ cm, $\mathbf{u}^* = (0.5, 0, 0, 0.063, 0)$, $f^* = 300.6$ cm². **4.80** $R^* = 7.871686 \times 10^{-2}$, $t^* = 1.574337 \times 10^{-3}$, $\mathbf{u}^* = (0, 3.056 \times 10^{-4}, 0.3038, 0, 0)$, $f^* = 30.56$ kg. **4.81** $R_0^* = 7.950204 \times 10^{-2}$, $R_1^* = 7.792774 \times 10^{-2}$, $\mathbf{u}^* = (0, 3.056 \times 10^{-4}, 0.3055, 0, 0)$, $f^* = 30.56$ kg. **4.82** $x_1^* = 60.50634$, $x_2^* = 1.008439$, $u_1^* = 19,918$, $u_2^* = 23,186$, $u_3^* = u_4^* = 0$, $f^* = 23,186.4$. **4.83** $h^* = 14$ m, $A^* = 5000$ m², $u_1^* = 5.9 \times 10^{-4}$, $u_2^* = 6.8 \times 10^{-4}$, $u_3^* = u_4^* = u_5^* = 0$, $f^* = \$13.4$ million. **4.84** $A^* = 20,000$, $B^* = 10,000$, $u_1^* = 35$, $u_3^* = 27$ (or, $u_1^* = 8$, $u_2^* = 108$), $f^* = -\$1,240,000$. **4.85** $R^* = 20$ cm, $H^* = 7.161973$ cm, $u_1^* = 10$, $u_3^* = 450$, $f^* = -9000$ cm³. **4.86** $R^* = 0.5$ cm, $N = 2546.5$, $u_1^* = 16,000$, $u_2^* = 4$, $f^* = -8000$ cm². **4.87** $W^* = 70.7107$ m, $D^* = 141.4214$ m, $u_3^* = 1.41421$, $u_4^* = 0$, $f^* = \$28,284.28$. **4.88** $A^* = 70$ kg, $B = 76$ kg, $u_1^* = 0.4$, $u_4^* = 16$, $f^* = -\$1308$. **4.89** $B^* = 0$, $M^* = 2.5$ kg, $u_1^* = 0.5$, $u_3^* = 1.5$, $f^* = \$2.5$. **4.90** $x_1^* = 316.667$, $x_2^* = 483.33$, $u_1^* = \frac{2}{3}$, $u_2^* = \frac{10}{3}$, $f^* = -\$1283.333$. **4.91** $r^* = 4.57078$ cm, $h^* = 9.14156$ cm, $\nu_1^* = -0.364365$, $u_1^* = 43.7562$, $f^* = 328.17$ cm². **4.92** $b^* = 10$ m, $h^* = 18$ m, $u_1^* = 0.04267$, $u_2^* = 0.00658$, $f^* = 0.545185$. **4.94** $D^* = 5.758823$ m, $H^* = 5.758823$ m, $\nu_1^* = -277.834$, $f^* = \$62,512.75$. **4.96** $P_1^* = 30.4$, $P_2^* = 29.6$, $u_1^* = 59.8$, $f^* = \$1789.68$. **4.134** (i) $\pi \leq x \leq 2\pi$ (ii) $\frac{\pi}{2} \leq x \leq \frac{3\pi}{2}$. **4.135** Convex everywhere. **4.136** Not convex. **4.137** $S = \{\mathbf{x} | x_1 \geq -\frac{5}{3}, (x_1 + \frac{11}{12})^2 - 4x_2^2 - \frac{9}{16} \geq 0\}$. **4.138** Not convex. **4.139** Convex everywhere. **4.140** Convex if $C \geq 0$. **4.141** Fails convexity check. **4.142** Fails convexity check. **4.143** Fails convexity check. **4.144** Fails convexity check. **4.145** Fails convexity check. **4.146** Fails convexity check. **4.147** Convex. **4.148** Fails convexity check. **4.149** Convex. **4.150** Convex. **4.151** $18.43^\circ \leq \theta \leq 71.57^\circ$. **4.152** $\theta \geq 71.57^\circ$. **4.153** No solution. **4.154** $\theta \leq 18.43^\circ$.

Chapter 5 More on Optimum Design Concepts

5.4 $x_1^* = 2.1667$, $x_2^* = 1.8333$, $\nu^* = -0.1667$; isolated minimum. **5.9** (1.5088, 3.2720), $\nu^* = -17.15$; not a minimum point; (2.5945, -2.0198), $\nu^* = -1.439$; isolated local minimum; (-3.6300, -3.1754), $\nu^* = -23.288$; not a minimum point; (-3.7322, 3.0879), $\nu^* = -2.122$; isolated local minimum. **5.20** (0.816, 0.75), $\mathbf{u}^* = (0, 0, 0, 0)$; not a minimum point; (0.816, 0), $\mathbf{u}^* = (0, 0, 0, 3)$; not a minimum point; (0, 0.75), $\mathbf{u}^* = (0, 0, 2, 0)$; not a minimum point; (1.5073, 1.2317), $\mathbf{u}^* = (0, 0.9632, 0, 0)$; not a minimum point; (1.0339, 1.6550), $\mathbf{u}^* = (1.2067, 0, 0, 0)$; not a minimum point; (0, 0), $\mathbf{u}^* = (0, 0, 2, 3)$; isolated local minimum; (2, 0), $\mathbf{u}^* = (2, 0, 0, 7)$; isolated local minimum; (0, 2), $\mathbf{u}^* = (1.667, 0, 3.667, 0)$; isolated local minimum; (1.386, 1.538), $\mathbf{u}^* = (0.633, 0.626, 0, 0)$; isolated local minimum. **5.21** (2.0870, 1.7391), $u^* = 0$; isolated global minimum. **5.22** $\mathbf{x}^* = (2.5, 1.5)$, $u^* = 1$, $f^* = 1.5$. **5.23** $\mathbf{x}^* = (6.3, 1.733)$, $\mathbf{u}^* = (0, 0.8, 0, 0)$, $f^* = -56.901$. **5.24** $\mathbf{x}^* = (1, 1)$, $u^* = 0$, $f^* = 0$. **5.25** $\mathbf{x}^* = (1, 1)$, $\mathbf{u}^* = (0, 0)$, $f^* = 0$. **5.26** $\mathbf{x}^* = (2, 1)$, $\mathbf{u}^* = (0, 2)$, $f^* = 1$. **5.27** (2.5945, 2.0198), $u^* = 1.4390$; isolated local minimum; (-3.6300, 3.1754), $u^* = 23.288$; not a minimum; (1.5088, -3.2720), $u^* = 17.150$; not a minimum; (-3.7322, -3.0879), $u^* = 2.122$; isolated local minimum. **5.28** (3.25, 0.75), $u^* = 0.75$, $\nu^* = -1.25$; isolated global minimum. **5.29** (2.3094, 0.3333), $u^* = 0$; not a minimum; (-2.3094, 0.3333), $u^* = 0$; not a minimum; (0, 3), $u^* = 16$; not a minimum; (2, 1), $u^* = 4$; isolated local minimum. **5.30** (-0.2857, -0.8571), $u^* = 0$; isolated local minimum. **5.38** $R_o^* = 20$ cm, $R_i^* = 19.84$ cm, $f^* = 79.1$ kg, $\mathbf{u}^* = (3.56 \times 10^{-3}, 0, 5.29, 0, 0, 0)$. **5.39** Multiple optima between (31.83, 1.0) and (25.23, 1.26) mm, $f^* = 45.9$ kg. **5.40** $R^* = 1.0077$ m, $t^* = 0.0168$ m, $f^* = 8182.8$ kg, $\mathbf{u}^* = (0.0417, 0.00408, 0, 0, 0)$. **5.41** $R^* = 0.0787$ m, $t^* = 0.00157$ m, $f^* = 30.56$ kg. **5.42** $R_o^* = 0.0795$ m, $R_i^* = 0.0779$ m, $f^* = 30.56$ kg. **5.43** $H^* = 8$ cm, $D^* = 7.98$ cm, $f^* = 300.6$ cm². **5.44** $A^* = 5000$ m², $h^* = 14$ m, $f^* = \$13.4$ million. **5.45** $x_1^* = 102.98$ mm, $x_2^* = 0.9546$, $f^* = 2.9$ kg, $\mathbf{u}^* = (4.568 \times 10^{-3}, 0, 3.332 \times 10^{-8}, 0, 0, 0, 0)$. **5.46** $d_o^* = 103$ mm, $d_i^* = 98.36$ mm, $f^* = 2.9$ kg, $\mathbf{u}^* = (4.657 \times 10^{-3}, 0, 3.281 \times 10^{-8}, 0, 0, 0, 0)$. **5.47** $R^* = 50.3$ mm, $t^* = 2.34$ mm, $f^* = 2.9$ kg, $\mathbf{u}^* = (4.643 \times 10^{-3}, 0, 3.240 \times 10^{-8}, 0, 0, 0, 0)$. **5.48** $H^* = 50$ cm, $D^* = 3.42$ cm, $f^* = 6.6$ kg, $\mathbf{u}^* = (0, 9.68 \times 10^{-5}, 0, 4.68 \times 10^{-2}, 0, 0)$. **5.50** Not a convex programming problem; $D^* = 10$ m, $H^* = 10$ m, $f^* = 60,000 \pi$ m³; $\Delta f = 800 \pi$ m³. **5.51** Convex; $A_1^* = 2.937 \times 10^{-4}$ m², $A_2^* = 6.556 \times 10^{-5}$ m², $f^* = 7.0$ kg. **5.52** $h^* = 14$ m, $A^* = 5000$ m², $u_1^* = 5.9 \times 10^{-4}$, $u_2^* = 6.8 \times 10^{-4}$, $u_3^* = u_4^* = u_5^* = 0$, $f^* = \$13.4$ million. **5.53** $R^* = 20$, $H^* = 7.16$, $u_1^* = 10$, $u_3^* = 450$, $f^* = -9000$ cm³. **5.54** $R^* = 0.5$ cm, $N^* = 2546.5$, $u_1^* = 16,022$, $u_2^* = 4$, $f^* = 8000$ cm². **5.55** $W^* = 70.7107$, $D^* = 141.4214$, $u_3^* = 1.41421$, $u_4^* = 0$, $f^* = \$28,284.28$. **5.56** $r^* = 4.57078$ cm, $h^* = 9.14156$ cm, $\nu_1^* = -0.364365$, $u_1^* = 43.7562$, $f^* = 328.17$ cm². **5.57** $b^* = 10$ m, $h^* = 18$ m, $u_1^* = 0.04267$, $u_2^* = 0.00658$, $f^* = 0.545185$. **5.58** $D^* = 5.758823$ m, $H^* = 5.758823$ m, $\nu_1^* = -277.834$, $f^* = \$62,512.75$. **5.59** $P_1^* = 30.4$, $P_2^* = 29.6$, $u_1^* = 59.8$, $f^* = \$1789.68$. **5.60** $R_o^* = 20$ cm, $R_i^* = 19.84$ cm, $f^* = 79.1$ kg. **5.61** Multiple optima between (31.83, 1.0) and (25.23, 1.26) mm, $f^* = 45.9$ kg. **5.62** $R^* = 0.0787$ m, $t^* = 0.00157$ m, $\mathbf{u}^* = (0, 3.056 \times 10^{-4}, 0.3038, 0, 0)$, $f^* = 30.56$ kg. **5.63** $R_o^* = 0.0795$ m, $R_i^* = 0.0779$ m, $\mathbf{u}^* = (0, 3.056 \times 10^{-4}, 0.3055, 0, 0)$, $f^* = 30.56$ kg. **5.64** $D^* = 7.98$ cm, $H^* = 8$ cm, $\mathbf{u}^* = (0.5, 0, 0, 0.063, 0)$, $f^* = 300.6$ cm². **5.65** $R^* = 1.0077$ m, $t^* = 0.0168$ m, $f^* = 8182.8$ kg, $\mathbf{u}^* = (0.0417, 0.00408, 0, 0, 0, 0)$. **5.66** $x_1^* = 102.98$ mm, $x_2^* = 0.9546$, $f^* = 2.9$ kg, $\mathbf{u}^* = (4.568 \times 10^{-3}, 0, 3.332 \times 10^{-8}, 0, 0, 0, 0)$. **5.67** $d_o^* = 103$ mm, $d_i^* = 98.36$ mm, $f^* = 2.9$ kg, $\mathbf{u}^* = (4.657 \times 10^{-3}, 0, 3.281 \times 10^{-8}, 0, 0, 0, 0)$. **5.68** $R^* = 50.3$ mm, $t^* = 2.34$, $f^* = 2.9$ kg, $\mathbf{u}^* = (4.643 \times 10^{-3}, 0, 3.240 \times 10^{-8}, 0, 0, 0, 0)$. **5.69** $R^* = 33.7$ mm, $t^* = 5.0$ mm, $f^* = 41.6$ kg, $\mathbf{u}^* = (0, 2.779 \times 10^{-4}, 0, 0, 0, 5.54, 0)$. **5.70** $R^* = 21.3$ mm, $t^* = 5.0$ mm, $f^* = 26.0$ kg, $\mathbf{u}^* = (0, 1.739 \times 10^{-4}, 0, 0, 0, 3.491, 0)$. **5.71** $R^* = 27.0$ mm, $t^* = 5.0$ mm, $f^* = 33.0$ kg, $\mathbf{u}^* = (0, 2.165 \times 10^{-4}, 0, 0, 0, 4.439, 0)$. **5.72** $A_1^* = 413.68$ mm², $A_2^* = 163.7$ mm², $f^* = 5.7$ kg, $\mathbf{u}^* = (0, 1.624 \times 10^{-2}, 0, 6.425 \times 10^{-3}, 0)$.

5.73 Multiple solutions $R^* = 20.0\text{mm}$, $t^* = 3.3\text{mm}$, $f^* = 8.1\text{kg}$, $\mathbf{u}^* = (0.0326, 0, 0, 0, 0, 0)$. **5.74** $A^* = 390\text{mm}^2$, $h^* = 500\text{mm}$, $f^* = 5.5\text{kg}$, $\mathbf{u}^* = (2.216 \times 10^{-2}, 0, 0, 0, 0, 1.67 \times 10^{-3})$. **5.75** $A^* = 415\text{mm}^2$, $s^* = 1480\text{mm}$, $f^* = 8.1\text{kg}$, $u_1^* = 0.0325$, all others are zero. **5.76** $A_1^* = 300\text{mm}^2$, $A_2^* = 50\text{mm}^2$, $f^* = 7.04\text{kg}$, $\mathbf{u}^* = (0.0473, 0, 0, 0, 0, 0, 0)$. **5.77** $R^* = 130\text{cm}$, $t^* = 2.86\text{cm}$, $f^* = 57,000\text{kg}$, $\mathbf{u}^* = (28170, 0, 294, 0, 0, 0, 0)$. **5.78** $d_o^* = 41.6\text{cm}$, $d_i^* = 40.2\text{cm}$, $f^* = 680\text{kg}$, $\mathbf{u}^* = (0, 0, 35.7, 6.1, 0, \dots)$. **5.79** $d_o^* = 1310\text{mm}$, $t^* = 14.2\text{mm}$, $f^* = 92,500\text{N}$, $\mathbf{u}^* = (0, 508, 462, 0, \dots)$. **5.80** $H^* = 50.0\text{cm}$, $D^* = 3.42\text{cm}$, $f^* = 6.6\text{kg}$, $\mathbf{u}^* = (0, 9.68 \times 10^{-5}, 0, 4.68 \times 10^{-2}, 0, 0)$.

Chapter 6 Linear Programming Methods for Optimum Design

6.21 $(0, 4, -3, -5)$; $(2, 0, 3, 1)$; $(1, 2, 0, -2)$; $(\frac{5}{3}, \frac{2}{3}, 2, 0)$. **6.22** $(0, 0, -3, -5)$; $(0, 1, 0, -3)$; $(0, 2.5, 4.5, 0)$; $(-3, 0, 0, -11)$; $(2.5, 0, -5.5, 0)$; $(\frac{9}{8}, \frac{11}{8}, 0, 0)$. **6.23** Decompose x_2 into two variables; $(0, 0, 0, 12, -3)$; $(0, 0, -3, 0, 6)$; $(0, 0, -1, 8, 0)$; $(0, 3, 0, 0, 6)$; $(0, 1, 0, 8, 0)$; $(4, 0, 0, 0, 1)$; $(3, 0, 0, 3, 0)$; $(4.8, 0, 0.6, 0, 0)$; $(4.8, -0.6, 0, 0, 0)$. **6.24** $(0, -\frac{8}{3}, -\frac{1}{3})$; $(2, 0, 3)$; $(0.2, -2.4, 0)$. **6.25** $(0, 0, 9, 2, 3)$; $(0, 9, 0, 20, -15)$; $(0, -1, 10, 0, 5)$; $(0, 1.5, 7.5, 5, 0)$; $(4.5, 0, 0, -2.5, 16.5)$; $(2, 0, 5, 0, 9)$; $(-1, 0, 11, 3, 0)$; $(4, 1, 0, 0, 13)$; $(\frac{15}{7}, \frac{33}{7}, 0, \frac{65}{7}, 0)$; $(-2.5, -2.25, 16.25, 0, 0)$. **6.26** $(0, 4, -3, -7)$; $(4, 0, 1, 1)$; $(3, 1, 0, -1)$; $(3.5, 0.5, 0.5, 0)$. **6.27** Decompose x_2 into two variables; 15 basic solutions; basic feasible solutions are $(0, 4, 0, 0, 7, 0)$; $(0, \frac{5}{3}, 0, \frac{7}{3}, 0, 0)$; $(2, 0, 0, 0, 1, 0)$; $(\frac{5}{3}, 0, 0, \frac{2}{3}, 0, 0)$; $(\frac{7}{3}, 0, \frac{2}{3}, 0, 0, 0)$. **6.28** Ten basic solutions; basic feasible solutions are $(2.5, 0, 0, 0, 4.5)$; $(1.6, 1.8, 0, 0, 0)$. **6.29** $(0, 0, 4, -2)$; $(0, 4, 0, 6)$; $(0, 1, 3, 0)$; $(-2, 0, 0, -4)$; $(2, 0, 8, 0)$; $(-1.2, 1.6, 0, 0)$. **6.30** $(0, 0, 0, -2)$; $(0, 2, -2, 0)$; $(0, 0, 0, -2)$; $(2, 0, 2, 0)$; $(0, 0, 0, -2)$; $(1, 1, 0, 0)$. **6.31** $(0, 0, 10, 18)$; $(0, 5, 0, 8)$; $(0, 9, -8, 0)$; $(-10, 0, 0, 48)$; $(6, 0, 16, 0)$; $(2, 6, 0, 0)$. **6.32** $\mathbf{x}^* = (\frac{10}{3}, 2)$; $f^* = -\frac{13}{3}$. **6.33** Infinite solutions between $\mathbf{x}^* = (0, 3)$ and $\mathbf{x}^* = (2, 0)$; $f^* = 6$. **6.34** $\mathbf{x}^* = (2, 4)$; $f^* = 10$. **6.35** $\mathbf{x}^* = (6, 0)$; $z^* = 12$. **6.36** $\mathbf{x}^* = (3.667, 1.667)$; $z^* = 15$. **6.37** $\mathbf{x}^* = (0, 5)$; $f^* = -5$. **6.55** $\mathbf{x}^* = (2, 4)$; $z^* = 10$. **6.56** Unbounded. **6.57** $\mathbf{x}^* = (3.5, 0.5)$; $z^* = 5.5$. **6.58** $\mathbf{x}^* = (1.667, 0.667)$; $z^* = 4.333$. **6.60** $\mathbf{x}^* = (0, 1.667, 2.333)$; $f^* = 4.333$. **6.61** $\mathbf{x}^* = (1.125, 1.375)$; $f^* = 36$. **6.62** $\mathbf{x}^* = (2, 0)$; $f^* = 40$. **6.63** $\mathbf{x}^* = (1.3357, 0.4406, 0, 3.2392)$; $z^* = 9.7329$. **6.64** $\mathbf{x}^* = (0.6541, 0.0756, 0.3151)$; $f^* = 9.7329$. **6.65** $\mathbf{x}^* = (0, 25)$; $z^* = 150$. **6.66** $\mathbf{x}^* = (\frac{2}{3}, \frac{5}{3})$; $z^* = \frac{16}{3}$. **6.67** $\mathbf{x}^* = (\frac{7}{3}, -\frac{2}{3})$; $z^* = -\frac{1}{3}$. **6.68** $\mathbf{x}^* = (1, 1)$; $f^* = 5$. **6.69** $\mathbf{x}^* = (2, 2)$; $f^* = 10$. **6.70** $\mathbf{x}^* = (4.8, -0.6)$; $z^* = 3.6$. **6.71** $\mathbf{x}^* = (2, 4)$; $z^* = 10$. **6.72** $\mathbf{x}^* = (0, 5)$; $z^* = 40$. **6.73** Infeasible problem. **6.74** Infinite solutions; $f^* = 0$. **6.77** $A^* = 20,000$, $B^* = 10,000$, Profit = \$1,240,000. **6.78** $A^* = 70$, $B^* = 76$, Profit = \$1308. **6.79** Bread = 0, Milk = 2.5 kg; Cost = \$2.5. **6.80** Bottles of wine = 316.67, Bottles of whiskey = 483.33; Profit = \$1283.3. **6.81** Shortening produced = 149,499.5 kg, Salad oil produced = 50,000 kg, Margarine produced = 10,000 kg; Profit = \$19,499.2. **6.82** $A^* = 10$, $B^* = 0$, $C^* = 20$; Capacity = 477,000. **6.83** $x_1^* = 0$, $x_2^* = 0$, $x_3^* = 200$, $x_4^* = 100$; $f^* = 786$. **6.84** $f^* = 1,333,679$ ton. **6.85** $\mathbf{x}^* = (0, 800, 0, 500, 1500, 0)$; $f^* = 7500$; $\mathbf{x}^* = (0, 0, 4500, 4000, 3000, 0)$; $f^* = 7500$; $\mathbf{x}^* = (0, 8, 0, 5, 15, 0)$, $f^* = 7500$. **6.86** (a) No effect (b) Cost decreases by 120,000. **6.87** 1. No effect; 2. Out of range, re-solve the problem; $A^* = 70$, $B^* = 110$; Profit = \$1580; 3. Profit reduces by \$4; 4. Out of range, re-solve the problem; $A^* = 41.667$, $B^* = 110$; Profit = \$1213.33. **6.88** $y_1 = 0.25$, $y_2 = 1.25$, $y_3 = 0$, $y_4 = 0$. **6.89** Unbounded. **6.90** $y_1 = 0$, $y_2 = 2.5$, $y_3 = -1.5$. **6.91** $y_1 = 0$, $y_2 = \frac{5}{3}$, $y_3 = -\frac{7}{3}$. **6.92** $y_1 = 4$, $y_2 = -1$. **6.93** $y_1 = -\frac{5}{3}$, $y_2 = -\frac{2}{3}$. **6.94** $y_1 = 2$, $y_2 = -6$. **6.95** $y_1 = 0$, $y_2 = 5$. **6.96** $y_1 = 0.654$, $y_2 = -0.076$, $y_3 = 0.315$. **6.97** $y_1 = -1.336$, $y_2 = -0.441$, $y_3 = 0$, $y_4 = -3.239$. **6.98** $y_1 = 0$, $y_2 = 0$, $y_3 = 0$, $y_4 = 6$. **6.99** $y_1 = -1.556$, $y_2 = 0.556$. **6.100** $y_1 = 0$, $y_2 = \frac{5}{3}$, $y_3 = -\frac{7}{3}$. **6.101** $y_1 = -0.5$, $y_2 = -2.5$. **6.102** $y_1 = -\frac{1}{3}$, $y_2 = 0$, $y_3 = \frac{5}{3}$. **6.103** $y_1 = 0.2$, $y_2 = 0.4$. **6.104** $y_1 = 0.25$, $y_2 = 1.25$, $y_3 = 0$, $y_4 = 0$. **6.105** $y_1 = 2$, $y_2 = 0$. **6.106** Infeasible problem. **6.107** $y_1 = 3$, $y_2 = 0$. **6.110** For $b_1 = 10$: $-8 \leq \Delta_1 \leq 8$; for $b_2 = 6$: $-2.667 \leq \Delta_2 \leq 8$; for $b_3 = 2$: $-4 \leq \Delta_3 \leq \infty$; for $b_4 = 6$: $-\infty \leq \Delta_4 \leq 8$. **6.111** Unbounded problem. **6.112** For $b_1 = 5$: $-0.5 \leq \Delta_1 \leq \infty$; for $b_2 = 4$: $-1 \leq \Delta_2 \leq 0.333$; for $b_3 = 3$: $-1 \leq \Delta_3 \leq 1$. **6.113** For $b_1 = 5$: $-2 \leq \Delta_1 \leq \infty$; for $b_2 = -4$:

$-2 \leq \Delta_2 \leq 2$; for $b_3 = 1$: $-2 \leq \Delta_1 \leq 1$. **6.114** For $b_1 = -5$: $-\infty \leq \Delta_1 \leq 4$; for $b_2 = -2$: $-8 \leq \Delta_2 \leq 4.5$. **6.115** $b_1 = 1$: $-5 \leq \Delta_1 \leq 7$; for $b_2 = 4$: $-3.5 \leq \Delta_2 \leq \infty$. **6.116** For $b_1 = -3$: $-4.5 \leq \Delta_1 \leq 5.5$; for $b_2 = 5$: $-3 \leq \Delta_2 \leq \infty$. **6.117** For $b_1 = 3$: $-\infty \leq \Delta_1 \leq 3$; for $b_2 = -8$: $-\infty \leq \Delta_2 \leq 4$. **6.118** For $b_1 = 8$: $-8 \leq \Delta_1 \leq \infty$; for $b_2 = 3$: $-14.307 \leq \Delta_2 \leq 4.032$; for $b_3 = 15$: $-20.16 \leq \Delta_3 \leq 101.867$. **6.119** For $b_1 = 2$: $-3.9178 \leq \Delta_1 \leq 1.1533$; for $b_2 = 5$: $-0.692 \leq \Delta_2 \leq 39.579$; for $b_3 = -4.5$: $-\infty \leq \Delta_3 \leq 7.542$; for $b_4 = 1.5$: $-2.0367 \leq \Delta_4 \leq 0.334$. **6.120** For $b_1 = 90$: $-15 \leq \Delta_1 \leq \infty$; for $b_2 = 80$: $-30 \leq \Delta_2 \leq \infty$; for $b_3 = 15$: $-\infty \leq \Delta_3 \leq 10$; for $b_4 = 25$: $-10 \leq \Delta_4 \leq 5$. **6.121** For $b_1 = 3$: $-1.2 \leq \Delta_1 \leq 15$; for $b_2 = 18$: $-15 \leq \Delta_2 \leq 12$. **6.122** For $b_1 = 5$: $-4 \leq \Delta_1 \leq \infty$; for $b_2 = 4$: $-7 \leq \Delta_2 \leq 2$; for $b_3 = 3$: $-1 \leq \Delta_3 \leq \infty$. **6.123** For $b_1 = 0$: $-2 \leq \Delta_1 \leq 2$; for $b_2 = 2$: $-2 \leq \Delta_2 \leq \infty$. **6.124** For $b_1 = 0$: $-6 \leq \Delta_1 \leq 3$; for $b_2 = 2$: $-\infty \leq \Delta_2 \leq 2$; for $b_3 = 6$: $-3 \leq \Delta_3 \leq \infty$. **6.125** For $b_1 = 12$: $-3 \leq \Delta_1 \leq \infty$; for $b_2 = 3$: $-\infty \leq \Delta_2 \leq 1$. **6.126** For $b_1 = 10$: $-8 \leq \Delta_1 \leq 8$; for $b_2 = 6$: $-2.667 \leq \Delta_2 \leq 8$; for $b_3 = 2$: $-4 \leq \Delta_3 \leq \infty$; for $b_4 = 6$: $-\infty \leq \Delta_4 \leq 8$. **6.127** For $b_1 = 20$: $-12 \leq \Delta_1 \leq \infty$; for $b_2 = 6$: $-\infty \leq \Delta_2 \leq 9$. **6.128** Infeasible problem. **6.129** For $b_1 = 0$: $-2 \leq \Delta_1 \leq 2$; for $b_2 = 2$: $-2 \leq \Delta_2 \leq \infty$. **6.132** For $c_1 = -1$: $-1 \leq \Delta c_1 \leq 1.667$; for $c_2 = -2$: $-\infty \leq \Delta c_2 \leq 1$. **6.133** Unbounded problem. **6.134** For $c_1 = -1$: $-\infty \leq \Delta c_1 \leq 3$; for $c_2 = -4$: $-3 \leq \Delta c_2 \leq \infty$. **6.135** For $c_1 = 1$: $-\infty \leq \Delta c_1 \leq 7$; for $c_2 = 4$: $-3.5 \leq \Delta c_2 \leq \infty$. **6.136** For $c_1 = 9$: $-5 \leq \Delta c_1 \leq \infty$; for $c_2 = 2$: $-9.286 \leq \Delta c_2 \leq 2.5$; for $c_3 = 3$: $-13 \leq \Delta c_3 \leq \infty$. **6.137** For $c_1 = 5$: $-2 \leq \Delta c_1 \leq \infty$; for $c_2 = 4$: $-2 \leq \Delta c_2 \leq 2$; for $c_3 = -1$: $0 \leq \Delta c_3 \leq 2$; for $c_4 = 1$: $0 \leq \Delta c_4 \leq \infty$. **6.138** For $c_1 = -10$: $-8 \leq \Delta c_1 \leq 16$; for $c_2 = -18$: $-\infty \leq \Delta c_2 \leq 8$. **6.139** For $c_1 = 20$: $-12 \leq \Delta c_1 \leq \infty$; for $c_2 = -6$: $-9 \leq \Delta c_2 \leq \infty$. **6.140** For $c_1 = 2$: $-3.918 \leq \Delta c_1 \leq 1.153$; for $c_2 = 5$: $-0.692 \leq \Delta c_2 \leq 39.579$; for $c_3 = -4.5$: $-\infty \leq \Delta c_3 \leq 7.542$; for $c_4 = 1.5$: $-3.573 \leq \Delta c_4 \leq 0.334$. **6.141** $c_1 = 8$: $-8 \leq \Delta c_1 \leq \infty$; for $c_2 = -3$: $-4.032 \leq \Delta c_2 \leq 14.307$; for $c_3 = 15$: $0 \leq \Delta c_3 \leq 101.8667$; for $c_4 = -15$: $0 \leq \Delta c_4 \leq \infty$. **6.142** For $c_1 = 10$: $-\infty \leq \Delta c_1 \leq 20$; for $c_2 = 6$: $-4 \leq \Delta c_2 \leq \infty$. **6.143** For $c_1 = -2$: $-\infty \leq \Delta c_1 \leq 2.8$; for $c_2 = 4$: $-5 \leq \Delta c_2 \leq \infty$. **6.144** For $c_1 = 1$: $-\infty \leq \Delta c_1 \leq 7$; for $c_2 = 4$: $-\infty \leq \Delta c_2 \leq 0$; for $c_3 = -4$: $-\infty \leq \Delta c_3 \leq 0$. **6.145** For $c_1 = 3$: $-1 \leq \Delta c_1 \leq \infty$; for $c_2 = 2$: $-5 \leq \Delta c_2 \leq 1$. **6.146** For $c_1 = 3$: $-5 \leq \Delta c_1 \leq 1$; for $c_2 = 2$: $-0.5 \leq \Delta c_2 \leq \infty$. **6.147** For $c_1 = 1$: $-0.3333 \leq \Delta c_1 \leq 0.5$; for $c_2 = 2$: $-\infty \leq \Delta c_2 \leq 0$; for $c_3 = -2$: $-1 \leq \Delta c_3 \leq 0$. **6.148** For $c_1 = 1$: $-1.667 \leq \Delta c_1 \leq 1$; for $c_2 = 2$: $-1 \leq \Delta c_2 \leq \infty$. **6.149** For $c_1 = 3$: $-\infty \leq \Delta c_1 \leq 3$; for $c_2 = 8$: $-4 \leq \Delta c_2 \leq 0$; for $c_3 = -8$: $-\infty \leq \Delta c_3 \leq 0$. **6.150** Infeasible problem. **6.151** For $c_1 = 3$: $0 \leq \Delta c_1 \leq \infty$; for $c_2 = -3$: $0 \leq \Delta c_2 \leq 6$. **6.154** For $c_1 = -48$: $-\infty \leq \Delta c_1 \leq 27$; for $c_2 = -28$: $-36 \leq \Delta c_2 \leq 4$. **6.155** For $c_1 = -10$: $-\infty \leq \Delta c_1 \leq 0.4$; for $c_2 = -8$: $-0.3333 \leq \Delta c_2 \leq 8$. **6.156** 1. $\Delta f = 0.5$; 2. $\Delta f = 0.5$ (Bread = 0, Milk = 3, $f^* = 3$); 3. $\Delta f = 0$. **6.157** 1. $\Delta f = 33.33$ (Wine bottles = 250, Whiskey bottles = 500, Profit = 1250); 2. $\Delta f = 63.33$. 3. $\Delta f = 83.33$ (Wine bottles = 400, Whiskey bottles = 400, Profit = 1200). **6.158** 1. Re-solve; 2. $\Delta f = 0$; 3. No change. **6.159** 1. Cost function increases by \$52.40; 2. No change; 3. Cost function increases by \$11.25, $x_1^* = 0$, $x_2^* = 30$, $x_3^* = 200$, $x_4^* = 70$. **6.160** 1. $\Delta f = 0$; 2. No change; 3. $\Delta f = 1800$ ($A^* = 6$, $B^* = 0$, $C^* = 22$, $f^* = -475,200$). **6.161** 1. $\Delta f = 0$; 2. $\Delta f = 2,485.65$; 3. $\Delta f = 0$; 4. $\Delta f = 14,033.59$; 5. $\Delta f = -162,232.3$. **6.162** 1. $\Delta f = 0$; 2. $\Delta f = 400$; 3. $\Delta f = -375$. **6.163** 1. $x_1^* = 0$, $x_2^* = 3$, $f^* = -12$; 2. $y_1 = \frac{4}{5}$, $y_2 = 0$; 3. $-15 \leq \Delta_1 \leq 3$, $-6 \leq \Delta_2 \leq \infty$; 4. $f^* = -14.4$, $b_1 = 18$.

Chapter 7 More on Linear Programming Methods for Optimum Design

7.1 $y_1^* = \frac{1}{4}$, $y_2^* = \frac{5}{4}$, $y_3^* = 0$, $y_4^* = 0$, $f_d^* = 10$. **7.2** Dual problem is infeasible. **7.3** $y_1^* = 0$, $y_2^* = 2.5$, $y_3^* = 1.5$, $f_d^* = 5.5$. **7.4** $y_1^* = 0$, $y_2^* = 1.6667$, $y_3^* = 2.3333$, $f_d^* = 4.3333$. **7.5** $y_1^* = 4$, $y_2^* = 1$, $f_d^* = -18$. **7.6** $y_1^* = 1.6667$, $y_2^* = 0.6667$, $f_d^* = -4.3333$. **7.7** $y_1^* = 2$, $y_2^* = 6$, $f_d^* = -36$. **7.8** $y_1^* = 0$, $y_2^* = 5$, $f_d^* = -40$. **7.9** $y_1^* = 0.65411$, $y_2^* = 0.075612$, $y_3^* = 0.315122$, $f_d^* = 9.732867$. **7.10** $y_1^* = 1.33566$, $y_2^* = 0.44056$, $y_3^* = 0$, $y_4^* = 3.2392$, $f_d^* = -9.732867$.

Chapter 8 Numerical Methods for Unconstrained Optimum Design

8.2 Yes. **8.3** No. **8.4** Yes. **8.5** No. **8.6** No. **8.7** No. **8.8** No. **8.9** Yes. **8.10** No. **8.11** No. **8.12** No. **8.13** No. **8.14** No. **8.16** $\alpha^* = 1.42850$, $f^* = 7.71429$.

8.17 $\alpha^* = 1.42758, f^* = 7.71429$. **8.18** $\alpha^* = 1.38629, f^* = 0.454823$. **8.19** \mathbf{d} is descent direction; slope = -4048 ; $\alpha^* = 0.15872$. **8.20** $\alpha^* = 0$. **8.21** $f(\alpha) = 4.1\alpha^2 - 5\alpha - 6.5$. **8.22** $f(\alpha) = 52\alpha^2 - 52\alpha + 13$. **8.23** $f(\alpha) = 6.88747 \times 10^9 \alpha^4 - 3.6111744 \times 10^8 \alpha^3 + 5.809444 \times 10^6 \alpha^2 - 27844\alpha + 41$. **8.24** $f(\alpha) = 8\alpha^2 - 8\alpha + 2$. **8.25** $f(\alpha) = 18.5\alpha^2 - 85\alpha - 13.5$. **8.26** $f(\alpha) = 288\alpha^2 - 96\alpha + 8$. **8.27** $f(\alpha) = 24\alpha^2 - 24\alpha + 6$. **8.28** $f(\alpha) = 137\alpha^2 - 110\alpha + 25$. **8.29** $f(\alpha) = 8\alpha^2 - 8\alpha$. **8.30** $f(\alpha) = 16\alpha^2 - 16\alpha + 4$. **8.31** $\alpha^* = 0.61$. **8.32** $\alpha^* = 0.5$. **8.33** $\alpha^* = 3.35\text{E-}03$. **8.34** $\alpha^* = 0.5$. **8.35** $\alpha^* = 2.2973$. **8.36** $\alpha^* = 0.16665$. **8.37** $\alpha^* = 0.5$. **8.38** $\alpha^* = 0.40145$. **8.39** $\alpha^* = 0.5$. **8.40** $\alpha^* = 0.5$. **8.41** $\alpha^* = 0.6097$. **8.42** $\alpha^* = 0.5$. **8.43** $\alpha^* = 3.45492\text{E-}03$. **8.44** $\alpha^* = 0.5$. **8.45** $\alpha^* = 2.2974$. **8.46** $\alpha^* = 0.1667$. **8.47** $\alpha^* = 0.5$. **8.48** $\alpha^* = 0.4016$. **8.49** $\alpha^* = 0.5$. **8.50** $\alpha^* = 0.5$. **8.52** $\mathbf{x}^{(2)} = (5/2, 3/2)$. **8.53** $\mathbf{x}^{(2)} = (0.1231, 0.0775)$. **8.54** $\mathbf{x}^{(2)} = (0.222, 0.0778)$. **8.55** $\mathbf{x}^{(2)} = (0.0230, 0.0688)$. **8.56** $\mathbf{x}^{(2)} = (0.0490, 0.0280)$. **8.57** $\mathbf{x}^{(2)} = (0.259, -0.225, 0.145)$. **8.58** $\mathbf{x}^{(2)} = (4.2680, 0.2244)$. **8.59** $\mathbf{x}^{(2)} = (3.8415, 0.48087)$. **8.60** $\mathbf{x}^{(2)} = (-1.590, 2.592)$. **8.61** $\mathbf{x}^{(2)} = (2.93529, 0.33976, 1.42879, 2.29679)$. **8.62** (8.52) $\mathbf{x}^* = (3.996096, 1.997073), f^* = -7.99999$; (8.53) $\mathbf{x}^* = (0.071659, 0.023233), f^* = -0.073633$; (8.54) $\mathbf{x}^* = (0.071844, -0.000147), f^* = -0.035801$; (8.55) $\mathbf{x}^* = (0.000011, 0.023273), f^* = -0.011626$; (8.56) $\mathbf{x}^* = (0.040028, 0.02501), f^* = -0.0525$; (8.57) $\mathbf{x}^* = (0.006044, -0.005348, 0.002467), f^* = 0.000015$; (8.58) $\mathbf{x}^* = (4.1453, 0.361605), f^* = -1616.183529$; (8.59) $\mathbf{x}^* = (3.733563, 0.341142), f^* = -1526.556493$; (8.60) $\mathbf{x}^* = (0.9087422, 0.8256927), f^* = 0.008348$, 1000 iterations; (8.61) $\mathbf{x}^* = (0.13189, 0.013188, 0.070738, 0.072022), f^* = 0.000409$, 1000 iterations. **8.63** $\mathbf{x}^* = (0.000023, 0.000023, 0.000045), f_1^* = 0$, 1 iteration; $\mathbf{x}^* = (0.002353, 0.0, 0.000007), f_2^* = 0.000006$, 99 iterations; $\mathbf{x}^* = (0.000003, 0.0, 0.023598), f_3^* = 0.000056$, 135 iterations. **8.64** Exact gradients are: 1. $\nabla f = (119.2, 258.0)$, 2. $\nabla f = (-202, 100)$, 3. $\nabla f = (6, 16, 16)$. **8.65** $\mathbf{u} = \frac{c}{2v}$, v = Lagrange multiplier for the equality constraints. **8.67** $\mathbf{x}^{(2)} = (4.2)$. **8.68** $\mathbf{x}^{(2)} = (0.07175, 0.02318)$. **8.69** $\mathbf{x}^{(2)} = (0.072, 0.0)$. **8.70** $\mathbf{x}^{(2)} = (0.0, 0.0233)$. **8.71** $\mathbf{x}^{(2)} = (0.040, 0.025)$. **8.72** $\mathbf{x}^{(2)} = (0.257, -0.229, 0.143)$. **8.73** $\mathbf{x}^{(2)} = (4.3682, 0.1742)$. **8.74** $\mathbf{x}^{(2)} = (3.7365, 0.2865)$. **8.75** $\mathbf{x}^{(2)} = (-1.592, 2.592)$. **8.76** $\mathbf{x}^{(2)} = (3.1134, 0.32224, 1.34991, 2.12286)$.

Chapter 9 More on Numerical Methods for Unconstrained Optimum Design

9.1 $\alpha^* = 1.42857, f^* = 7.71429$. **9.2** $\alpha^* = \frac{10}{7}, f^* = 7.71429$, one iteration. **9.4** 1. $\alpha^* = \frac{13}{4}$. 2. $\alpha = 1.81386$ or 4.68614 . **9.10** $\mathbf{x}^{(1)} = (4, 2)$. **9.11** $\mathbf{x}^{(1)} = (0.071598, 0.023251)$. **9.12** $\mathbf{x}^{(1)} = (0.071604, 0.0)$. **9.13** $\mathbf{x}^{(1)} = (0.0, 0.0232515)$. **9.14** $\mathbf{x}^{(1)} = (0.04, 0.025)$. **9.15** $\mathbf{x}^{(1)} = (0, 0, 0)$. **9.16** $\mathbf{x}^{(1)} = (-2.7068, 0.88168)$. **9.17** $\mathbf{x}^{(1)} = (3.771567, 0.335589)$. **9.18** $\mathbf{x}^{(1)} = (4.99913, 24.99085)$. **9.19** $\mathbf{x}^{(1)} = (-1.26859, -0.75973, 0.73141, 0.39833)$. **9.22** $\mathbf{x}^{(2)} = (4, 2)$. **9.23** $\mathbf{x}^{(2)} = (0.0716, 0.02325)$. **9.24** $\mathbf{x}^{(2)} = (0.0716, 0.0)$. **9.25** $\mathbf{x}^{(2)} = (0.0, 0.02325)$. **9.26** $\mathbf{x}^{(2)} = (0.04, 0.025)$. **9.27** DFP: $\mathbf{x}^{(2)} = (0.2571, -0.2286, 0.1428)$; BFGS: $\mathbf{x}^{(2)} = (0.2571, -0.2286, 0.1429)$. **9.28** DFP: $\mathbf{x}^{(2)} = (4.37045, 0.173575)$; BFGS: $\mathbf{x}^{(2)} = (4.37046, 0.173574)$. **9.29** $\mathbf{x}^{(2)} = (3.73707, 0.28550)$. **9.30** $\mathbf{x}^{(2)} = (-1.9103, -1.9078)$. **9.31** DFP: $\mathbf{x}^{(2)} = (3.11339, 0.32226, 1.34991, 2.12286)$; BFGS: $\mathbf{x}^{(2)} = (3.11339, 0.32224, 1.34991, 2.12286)$. **9.44** $x_1 = 3.7754$ mm, $x_2 = 2.2835$ mm. **9.45** $x_1 = 2.2213$ mm, $x_2 = 1.8978$ mm. **9.46** $x^* = 0.619084$. **9.47** $x^* = 9.424753$. **9.48** $x^* = 1.570807$. **9.49** $x^* = 1.496045$. **9.50** $\mathbf{x}^* = (3.667328, 0.739571)$. **9.51** $\mathbf{x}^* = (4.000142, 7.999771)$.

Chapter 10 Numerical Methods for Constrained Optimum Design

10.29 $\mathbf{x}^* = (\frac{5}{2}, \frac{5}{2}), u^* = 1, f^* = 0.5$. **10.30** $\mathbf{x}^* = (1, 1), u^* = 0, f^* = 0$. **10.31** $\mathbf{x}^* = (\frac{4}{5}, \frac{3}{5}), u^* = \frac{2}{5}, f^* = \frac{1}{5}$. **10.32** $\mathbf{x}^* = (2, 1), u^* = 0, f^* = -3$. **10.33** $\mathbf{x}^* = (1, 2), u^* = 0, f^* = -1$. **10.34** $\mathbf{x}^* = (\frac{13}{6}, \frac{11}{6}), v^* = -\frac{1}{6}, f^* = -\frac{25}{3}$. **10.35** $\mathbf{x}^* = (3, 1), v_1^* = -2, v_2^* = -2, f^* = 2$. **10.36** $\mathbf{x}^* = (\frac{48}{23}, \frac{40}{23}), u^* = 0, f^* = -\frac{192}{23}$. **10.37** $\mathbf{x}^* = (\frac{5}{2}, \frac{3}{2}), u^* = 1, f^* = -\frac{9}{2}$. **10.38** $\mathbf{x}^* = (\frac{63}{10}, \frac{26}{15}), u_1^* = 0, u_2^* = \frac{4}{5}, f^* = -\frac{3547}{50}$. **10.39** $\mathbf{x}^* = (2, 1), u_1^* = 0, u_2^* = 2, f^* = -1$. **10.40** $\mathbf{x}^* = (0.241507, 0.184076, 0.574317); \mathbf{u}^* = (0, 0, 0, 0), v_1^* = -0.7599, f^* = 0.3799$.

Chapter 12 Introduction to Optimum Design with MATLAB

12.1 For $l = 0.5$ m, $T_o = 10$ kN·m, $T_{\max} = 20$ kN·m, $x_1^* = 103$ mm, $x_2^* = 0.955$, $f^* = 2.9$ kg. **12.2** For $l = 0.5$, $T_o = 10$ kN·m, $T_{\max} = 20$ kN·m, $d_o^* = 103$ mm, $d_i^* = 98.36$ mm, $f^* = 2.9$ kg. **12.3** $R^* = 50.3$ mm, $t^* = 2.35$ mm, $f^* = 2.9$ kg. **12.4** $A_1^* = 300$ mm², $A_2^* = 50$ mm², $f^* = 7$ kg. **12.5** $R^* = 130$ cm, $t^* = 2.86$ cm, $f^* = 57,000$ kg. **12.6** $d_o^* = 41.56$ cm, $d_i^* = 40.19$ cm, $f^* = 680$ kg. **12.7** $d_o^* = 1310$ mm, $t^* = 14.2$ mm, $f^* = 92,500$ N. **12.8** $H^* = 50.0$ cm, $D^* = 3.42$ cm, $f^* = 6.6$ kg. **12.9** $b^* = 0.5$ in, $h^* = 0.28107$ in, $f^* = 0.140536$ in²; Active constraints: fundamental vibration frequency and lower limit on b . **12.10** $b^* = 50.4437$ cm, $h^* = 15.0$ cm, $t_1^* = 1.0$ cm, $t_2^* = 0.5218$ cm, $f^* = 16,307.2$ cm³; Active constraints: axial stress, shear stress, upper limit on t_1 and upper limit on h . **12.11** $A_1^* = 1.4187$ in², $A_2^* = 2.0458$ in², $A_3^* = 2.9271$ in², $x_1^* = -4.6716$ in, $x_2^* = 8.9181$ in, $x_3^* = 4.6716$ in, $f^* = 75.3782$ in³; Active stress constraints: member 1—loading condition 3, member 2—loading condition 1, member 3—loading conditions 1 and 3. **12.12** For $\phi = \sqrt{2}$: $x_1^* = 2.4138$, $x_2^* = 3.4138$, $x_3^* = 3.4141$, $f^* = 1.2877 \times 10^{-7}$; For $\phi = 2^{1/3}$: $x_1^* = 2.2606$, $x_2^* = 2.8481$, $x_3^* = 2.8472$, $f^* = 8.03 \times 10^{-7}$.

Chapter 13 Interactive Design Optimization

13.1 Several local minima: (0, 0), (2, 0), (0, 2), (1.386, 1.538). **13.2** $\mathbf{d} = (0, 0)$. **13.3** $\mathbf{d} = (-\frac{21}{13}, -\frac{19}{13})$, $f(\mathbf{x}^{(1)}) = -0.00364$. **13.4** Subproblem is infeasible. **13.5** $\mathbf{d} = (-0.01347, -0.7663)$. **13.6** $\mathbf{x}^* = (0.76471, 1.05882)$, $f(\mathbf{x}^*) = -4.05882$. **13.7** $\mathbf{d} = (1.5294, -0.38235)$; $f(\mathbf{x}^{(1)}) = -3.73362$. **13.8** $\mathbf{d} = (-1.2308, -1.8462)$; $f(\mathbf{x}^{(1)}) = 1.53846$. **13.9** $\mathbf{d} = (-0.2353, -0.9412)$; $f(\mathbf{x}^{(1)}) = -4.05822$. **13.10** $\mathbf{d} = (-0.29412, -1.1765)$; $f(\mathbf{x}^{(1)}) = -3.11765$.

Chapter 14 Design Optimization Applications with Implicit Functions

14.1 For $l = 500$ mm, $d_o^* = 102.985$ mm, $d_o^*/d_i^* = 0.954614$, $f^* = 2.900453$ kg; Active constraints: shear stress and critical torque. **14.2** For $l = 500$ mm, $d_o^* = 102.974$ mm, $d_i^* = 98.2999$ mm, $f^* = 2.90017$ kg; Active constraints: shear stress and critical torque. **14.3** For $l = 500$ mm, $R^* = 50.3202$ mm, $t^* = 2.33723$ mm, $f^* = 2.90044$ kg; Active constraints: shear stress and critical torque. **14.5** $R^* = 129.184$ cm, $t^* = 2.83921$ cm, $f^* = 56,380.61$ kg; Active constraints: combined stress and diameter/thickness ratio. **14.6** $d_o^* = 41.5442$ cm, $d_i^* = 40.1821$ cm, $f^* = 681.957$ kg; Active constraints: deflection and diameter/thickness ratio. **14.7** $d_o^* = 1308.36$ mm, $t^* = 14.2213$ mm, $f^* = 92,510.7$ N; Active constraints: diameter/thickness ratio and deflection. **14.8** $H^* = 50$ cm, $D^* = 3.4228$ cm, $f^* = 6.603738$ kg; Active constraints: buckling load and minimum height. **14.9** $b^* = 0.5$ in, $h^* = 0.28107$ in, $f^* = 0.140536$ in²; Active constraints: fundamental vibration frequency and lower limit on b . **14.10** $b^* = 50.4437$ cm, $h^* = 15.0$ cm, $t_1^* = 1.0$ cm, $t_2^* = 0.5218$ cm, $f^* = 16,307.2$ cm³; Active constraints: axial stress, shear stress, upper limit on t_1 and upper limit on h . **14.11** $A_1^* = 1.4187$ in², $A_2^* = 2.0458$ in², $A_3^* = 2.9271$ in², $x_1^* = -4.6716$ in, $x_2^* = 8.9181$ in, $x_3^* = 4.6716$ in, $f^* = 75.3782$ in³; Active stress constraints: member 1—loading condition 3, member 2—loading condition 1, member 3—loading conditions 1 and 3. **14.12** For $\phi = \sqrt{2}$: $x_1^* = 2.4138$, $x_2^* = 3.4138$, $x_3^* = 3.4141$, $f^* = 1.2877 \times 10^{-7}$; For $\phi = 2^{1/3}$: $x_1^* = 2.2606$, $x_2^* = 2.8481$, $x_3^* = 2.8472$, $f^* = 8.03 \times 10^{-7}$. **14.13** d_o^* at base = 48.6727 cm, d_o^* at top = 16.7117 cm, $t^* = 0.797914$ cm, $f^* = 623.611$ kg. **14.14** d_o^* at base = 1419 mm, d_o^* at top = 956.5 mm, $t^* = 15.42$ mm, $f^* = 90,894$ kg. **14.15** Outer dimension at base = 42.6407 cm, outer dimension at top = 14.6403 cm, $t^* = 0.699028$ cm, $f^* = 609.396$ kg. **14.16** Outer dimension at base = 1243.2 mm, outer dimension at top = 837.97 mm, $t^* = 13.513$ mm, $f^* = 88,822.2$ kg. **14.17** $u_a = 25$: $f_1 = 1.07301\text{E-}06$, $f_2 = 1.83359\text{E-}02$, $f_3 = 24.9977$; $u_a = 35$: $f_1 = 6.88503\text{E-}07$, $f_2 = 1.55413\text{E-}02$, $f_3 = 37.8253$. **14.18** $u_a = 25$: $f_1 = 2.31697\text{E-}06$, $f_2 = 2.74712\text{E-}02$, $f_3 = 7.54602$; $u_a = 35$: $f_1 = 2.31097\text{E-}06$, $f_2 = 2.72567\text{E-}02$, $f_3 = 7.48359$. **14.19** $u_a = 25$: $f_1 = 1.11707\text{E-}06$, $f_2 = 1.52134\text{E-}02$, $f_3 = 19.815$, $f_4 = 3.3052\text{E-}02$; $u_a = 3.5$: $f_1 = 6.90972\text{E-}07$, $f_2 = 1.36872\text{E-}02$, $f_3 = 31.479$, $f_4 = 2.3974\text{E-}02$. **14.20** $f_1 = 1.12618\text{E-}06$, $f_2 = 1.798\text{E-}02$, $f_3 = 33.5871$, $f_4 = 0.10$. **14.21** $f_1 = 2.34615\text{E-}06$, $f_2 = 2.60131\text{E-}02$, $f_3 = 10.6663$, $f_4 = 0.10$.

14.22 $f_1 = 1.15097\text{E-}06$, $f_2 = 1.56229\text{E-}02$, $f_3 = 28.7509$, $f_4 = 3.2547\text{E-}02$. **14.23** $f_1 = 8.53536\text{E-}07$, $f_2 = 1.68835\text{E-}02$, $f_3 = 31.7081$, $f_4 = 0.10$. **14.24** $f_1 = 2.32229\text{E-}06$, $f_2 = 2.73706\text{E-}02$, $f_3 = 7.48085$, $f_4 = 0.10$. **14.25** $f_1 = 8.65157\text{E-}07$, $f_2 = 1.4556\text{E-}02$, $f_3 = 25.9761$, $f_4 = 2.9336\text{E-}02$. **14.26** $f_1 = 8.27815\text{E-}07$, $f_2 = 1.65336\text{E-}02$, $f_3 = 28.2732$, $f_4 = 0.10$. **14.27** $f_1 = 2.313\text{E-}06$, $f_2 = 2.723\text{E-}02$, $f_3 = 6.86705$, $f_4 = 0.10$. **14.28** $f_1 = 8.39032\text{E-}07$, $f_2 = 1.43298\text{E-}2$, $f_3 = 25.5695$, $f_4 = 2.9073\text{E-}02$. **14.29** $k^* = 2084.08$, $c^* = 300$ (upper limit), $f^* = 1.64153$.

Chapter 18 Global Optimization Concepts and Methods for Optimum Design

18.1 Six local minima, two global minima: $(0.0898, -0.7126)$, $(-0.0898, 0.7126)$, $f_G^* = -1.0316258$. **18.2** 10^n local minima; global minimum: $x_i^* = 1$, $f_G^* = 0$. **18.3** Many local minima; global minimum: $\mathbf{x}^* = (0.195, -0.179, 0.130, 0.130)$, $f_G^* = 3.13019 \times 10^{-4}$. **18.4** Many local minima; two global minima: $\mathbf{x}^* = (0.05, 0.85, 0.65, 0.45, 0.25, 0.05)$, $\mathbf{x}^* = (0.55, 0.35, 0.15, 0.95, 0.75, 0.55)$, $f_G^* = -1$. **18.5** Local minima: $\mathbf{x}^* = (0, 0)$, $f^* = 0$; $\mathbf{x}^* = (0, 2)$, $f^* = -2$; $\mathbf{x}^* = (1.38, 1.54)$, $f^* = -0.04$; Global minimum: $\mathbf{x}^* = (2, 0)$, $f^* = -4$.

Appendix A Economic Analysis

A.7 (a) $S_{216} = \$24,004.31$, (b) $R = \$586.02$. **A.8** (a) $R = \$156.89$, (b) $P = \$2,293.37$. **A.11** (a) $R = \$843.53$, (b) $P = \$67,512.98$. **A.12** (a) $S_{24} = \$2,392.83$, (b) $S_{730} = \$2,394.38$. **A.13** $i = 0.02075$ (24.9% annual). **A.14** $\$855.01$. **A.16** $PW_A = \$598,935.18$, $PW_B = \$518,807.40$; $AC_A = \$60,408.07$, $AC_B = \$52,326.46$. **A.23** $PW_A = \$24 \times 10^6$, $PW_B = \$27.5 \times 10^6$. **A.24** $AC_{80} = \$53,134.42$, $AC_{50} = \$53,507.54$; $PW_{80} = \$343,468.81$, $PW_{50} = \$345,880.69$. **A.25** $PW_A = \$54,668.58$, $PW_B = \$54,889.53$. **A.26** $PW_A = \$101,600.00$, $PW_B = \$102,546.86$; $AC_A = \$14,187.96$ ($MC_A = \$1,118.70$), $AC_B = \$14,442.54$ ($MC_B = \$1,136.14$). **A.27** (a) $AC_A = -\$4,475.76$, $AC_B = -\$5,983.84$, $AC_C = -\$10,459.63$; $PW_A = -\$38,104.76$, $PW_B = -\$50,943.86$, $PW_C = -\$89,048.62$; (b) $A = \$213,296.47$, $B = \$171,977.88$, $C = \$295,933.65$. **A.28** $PW_A = \$106,567.11$, $PW_B = \$123,283.56$. **A.29** $AC_A = \$21,013.62$, $AC_B = \$24,000.00$; $PW_A = \$104,996.60$, $PW_B = \$119,918.35$. **A.30** $PW_A = \$1,186,767.70$, $PW_B = \$1,159,216.60$; $AC_A = \$118,676.77$, $AC_B = \$115,921.66$. **A.31** $PW_A = \$76,665.15$, $PW_B = \$100,000.00$, $PW_C = \$54,884.60$. **A.32** $PW_A = \$11,928.68$, $PW_B = \$12,708.79$, $PW_C = \$12,392.64$.

Appendix B Vector and Matrix Algebra

B.1 $|A| = 1$. **B.2** $|A| = 14$. **B.3** $|A| = -20$. **B.4** $\lambda_1 = (5 - \sqrt{5})/2$, $\lambda_2 = 2$, $\lambda_3 = (5 + \sqrt{5})/2$. **B.5** $\lambda_1 = (2 - \sqrt{2})$, $\lambda_2 = 2$, $\lambda_3 = (2 + \sqrt{2})$. **B.6** $r = 4$. **B.7** $r = 4$. **B.8** $r = 4$. **B.9** $x_1 = 1$, $x_2 = 1$, $x_3 = 1$. **B.10** $x_1 = 1$, $x_2 = 1$, $x_3 = 1$. **B.11** $x_1 = 1$, $x_2 = 2$, $x_3 = 3$. **B.12** $x_1 = 1$, $x_2 = 1$, $x_3 = 1$, $x_4 = 1$. **B.13** $x_1 = 1$, $x_2 = 2$, $x_3 = 3$. **B.14** $x_1 = 2$, $x_2 = 1$, $x_3 = 1$. **B.15** $x_1 = 1$, $x_2 = 2$, $x_3 = 3$. **B.16** $x_1 = 1$, $x_2 = 1$, $x_3 = 1$. **B.17** $x_1 = 2$, $x_2 = 1$, $x_3 = 1$, $x_4 = -2$. **B.18** $x_1 = 6$, $x_2 = -15$, $x_3 = -1$, $x_4 = 9$. **B.19** $x_1 = (3 - 7x_3 - 2x_4)/4$, $x_2 = (-1 + x_3 - 2x_4)/4$. **B.20** $x_1 = (4 - x_3)$, $x_2 = 2$, $x_4 = 4$. **B.21** $x_1 = (-3 - 4x_4)$, $x_2 = (-2 - 3x_4)$, $x_3 = x_4$. **B.22** $x_1 = -x_4$, $x_2 = (2 + x_4)$, $x_3 = (-2 - x_4)$. **B.27** $x_1 = 4 + (2x_2 - 8x_3 - 5x_4 + 2x_5)/3$; $x_6 = (9 - x_2 - 2x_3 - x_4 - 3x_5)/2$; $x_7 = 14 - (13x_2 + 14x_3 + 8x_4 + 4x_5)/3$; $x_8 = (9 + x_2 - 16x_3 - 7x_4 + 19x_5)/6$. **B.28** Linearly dependent. **B.29** Linearly independent. **B.30** $\lambda_1 = (3 - 2\sqrt{2})$, $\lambda_2 = (3 + 2\sqrt{2})$. **B.31** $\lambda_1 = (3 - \sqrt{5})$, $\lambda_2 = (3 + \sqrt{5})$. **B.32** $\lambda_1 = (5 - \sqrt{13})/2$, $\lambda_2 = (5 + \sqrt{13})/2$, $\lambda_3 = 5$. **B.33** $\lambda_1 = (1 - \sqrt{2})$, $\lambda_2 = 1$, $\lambda_3 = (1 + \sqrt{2})$. **B.34** $\lambda_1 = 0$, $\lambda_2 = (3 - \sqrt{5})$, $\lambda_3 = (3 + \sqrt{5})$.

Appendix C A Numerical Method for Solution of Nonlinear Equations

C.1 $x^{(3)} = 0.6190$. **C.2** $x^{(2)} = 9.4249$. **C.3** $x^{(2)} = 1.5708$. **C.4** $x^{(3)} = -1.4958$. **C.5** $\mathbf{x}^{(2)} = (3.78842, 0.6545)$. **C.6** $\mathbf{x}^{(2)} = (4.10321, 8.14637)$. **C.7** $\mathbf{x}^{(2)} = (-3.3738, 0.03760)$. **C.8** $\mathbf{x} = (3.6840, 0.7368)$. **C.9** $\mathbf{x} = (-4, -8)$; $\mathbf{x} = (4, 8)$. **C.10** $\mathbf{x} = (-3.3315, 0.03949)$; $\mathbf{x} = (-0.3980, 0.5404)$.

Index

A

- Acceptance criterion, 578
- Acceptance-rejection (A-R) method, 578, 586
- Algebra, vector and matrix, 611–646
 - concepts related to set of vectors, 635–641
 - definition of matrices, 611–613
 - eigenvalues and eigenvectors, 642–643
 - exercises, 645–646
 - norm and condition number of matrix, 643–644
 - solution of m linear equations in n unknowns, 628–635
 - type of matrices and their operations, 613–618
- Algorithms
 - attributes of good optimization, 478–479
 - conceptual local-global, 579–580
 - and constrained problems, 340–342
 - constraint correction, 440–442
 - for constraint correction at constant cost, 442–445
 - for constraint correction at specified increase in cost, 445
 - convergence of, 345–346
 - cost reduction, 436–440
 - CSD, 368
 - modified constrained steepest descent, 404–405
 - observations on interactive, 447–448
 - Phase I, 220
 - Phase II, 221
 - robust, 478
 - selection of, 478
 - Simplex, 211–212
- Algorithms, concepts related to numerical, 278–282
 - convergence of algorithms, 282
 - descent direction and descent step, 280–281
 - example—checking for descent condition, 281
 - general algorithm, 279–280
 - rate of convergence, 282
- Algorithms for step size determination, ideas and, 282–293
 - alternate equal interval search, 288–289
 - analytical method to compare step size, 283–285
 - definition of one-dimensional minimization subproblem, 282–283
 - equal interval search, 286–288
 - example—analytical step size determination, 284–285
 - example—minimization of function by golden section search, 292–293
 - golden section search, 289–293
 - numerical methods and compute step size, 285–286
- Algorithms, interactive design optimization, 436–448
 - algorithm for constraint correction at constant cost, 442–445
 - algorithm for constraint correction at specified increase in cost, 445
 - constraint correction algorithm, 440–442
 - cost reduction algorithm, 436–440
 - example—constraint correction at constant cost, 443–444, 444–445
 - example—constraint correction at specified increase in cost, 445

- Algorithms, interactive design optimization
 - (*continued*)
 - example—constraint correction step, 441–442
 - example—constraint correction with minimum increase in cost, 446–447
 - example—cost reduction step, 437–439
 - example—cost reduction step with potential constraints, 440
 - observations on interactive algorithms, 447–448
 - Algorithms, SLP, 352–358
 - basic idea—move limits, 352–353
 - example—sequential linear programming algorithm, 354–355
 - example—use of sequential linear programming, 356–357
 - SLP algorithm, 353–357
 - SLP algorithm observations, 357–358
 - Alternate equal interval search, 288–289
 - Alternate quadratic interpolation, 308–309
 - American Association of State Highway and Transportation Officials (AASHTO), 369
 - Analyses
 - engineering, 4
 - operations, 583–585
 - Analyses, economic, 593–609
 - economic bases for comparison, 598–604
 - exercises, 604–609
 - time value of money, 593–598
 - Analyses, postoptimality, 143–148, 228–243
 - changes in coefficient matrix, 241–243
 - changes in resource limits, 229–230
 - constraint variation sensitivity result, 148
 - effect of scaling constraint on Lagrange multiplier, 147–148
 - example— \leq type constraints, 236–237
 - example— \leq type constraints, 239–240
 - example—effect of scaling constraint, 147–148
 - example—effect of scaling cost function, 146–147
 - example—equality and \geq type constraints, 237–238, 240–241
 - example—Lagrange multipliers, 146–147, 147–148
 - example—optimum cost function, 145
 - example—ranges for cost coefficients, 239–240, 240–241
 - example—ranges for resource limits, 236–237, 237–238
 - example—recovery of Lagrange multipliers for \leq type constraint, 230–232
 - example—variations of constraint limits, 145
 - example—ranging cost coefficients, 239–241
 - example—ranging right side parameters, 235–238
 - example—recovery of Lagrange multipliers for \geq type constraints, 232–234
 - example—scaling cost function on Lagrange multipliers, 146–147
 - Annual base comparisons, 599–601
 - Annual cost (AC), 598
 - Applications, design optimization, 465–511
 - Array operation, 414
 - Artificial cost function, 219–220
 - Artificial variables, 218–228
 - artificial cost function, 219–220
 - definition of Phase I problem, 220
 - degenerate basic feasible solution, 226–228
 - example—feasible problem, 223–224
 - example—implications of degenerate feasible solution, 226–228
 - example—unbounded problem, 225–226
 - example—use of artificial variables, 225–226
 - example—use of artificial variables for \geq type constraints, 221–223
 - example—use of artificial variables for equality constraints, 223–224
 - Phase I algorithm, 220
 - Phase II algorithm, 221
 - use for equality constraints, 223–224
 - Ascents, alternation of descents and, 570
 - Asymmetric three-bar structure, 484–490
 - Augmented Lagrangian methods, 334
- B**
- Basic feasible solution, degenerate, 226–228
 - BBM. *See* Branch and bound method
 - Beam, design of rectangular, 162–166
 - Beam design problem, graphical solution for, 69–72
 - Binary variable defined, 513
 - Bounded objective function method, 558–559
 - Brackets
 - design of two-bar, 24–29, 28–29
 - design of wall, 158–162
 - Branch and bound method (BBM), 516–521
 - basic, 517–518
 - BBM for general MV-OPT, 520–521
 - BBM with local minimization, 519–520
 - example—BBM with local minimizations, 519–520
 - example—BBM with only discrete values allowed, 517–518
 - British versus SI units. *See* U.S.-British versus SI units
 - Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, 327–328

C

- Cabinet, design of, 30–32
- Calculus concepts, 89–103
 - concept of necessary and sufficient conditions, 102–103
 - example—calculation of gradient vector, 91
 - example—evaluation of gradient and Hessian of function, 92–93
 - example—linear Taylor’s expansion of function, 95–96
 - example—Taylor’s expansion of a function of one variable, 94
 - example—Taylor’s expansion of a function of two variables, 95
 - gradient vector, 90–91
 - Hessian matrix, 92–93
 - quadratic forms and definite matrices, 96–102
 - Taylor’s expansion, 93–96
- Can, design of, 18–20
- Canonical form/general solution of $Ax = b$, 202–203
- Capabilities, desired interactive, 448–450
- Cash flow diagrams, 594
- Changing constraint limits, effect of, 143–145
- Chromosome defined, 532
- Clustering methods, 573–575
- Coefficient matrix, changes in, 241–243
- Coefficients, ranging cost, 239–241
- Coil springs, design of, 36–38
- Column design
 - for minimum mass, 421–425
 - minimum weight tubular, 32–35
- Column matrix, 613
- Column vector, 613
- Columns, graphical solutions for minimum weight tubular, 69
- Comparisons
 - annual base, 599–601
 - PW, 601
- Comparisons, economic bases for, 598–604
 - alternate power stations, 602
 - annual base comparisons, 599–601
 - example—alternate designs, 599, 602
 - example—alternate power stations, 600
 - example—alternate quarries, 600–601, 603
 - PW comparisons, 601
- Compromise solution, 550
- Computer-aided design optimization (CADO), 5
- Computer programs, sample, 657–673
 - equal interval search, 657–660
 - golden section search, 660
 - modified Newton’s method, 669–673
 - steepest descent method, 660–669
- Computers, role in interactive design optimization, 434–435
- Concepts and methods, multiobjective optimum design, 543–563
- Concepts, calculus, 89–103
 - concept of necessary and sufficient conditions, 102–103
 - example—calculation of gradient vector, 91
 - example—evaluation of gradient and Hessian of function, 92–93
 - example—linear Taylor’s expansion of function, 95–96
 - example—Taylor’s expansion of a function of one variable, 94
 - example—Taylor’s expansion of a function of two variables, 95
 - gradient vector, 90–91
 - Hessian matrix, 92–93
 - quadratic forms and definite matrices, 96–102
 - Taylor’s expansion, 93–96
- Concepts, optimum design, 83–174, 175–190
 - alternate form of KKT necessary conditions, 175–178
 - basic calculus concepts, 89–103
 - constrained optimum design problems, 119–143
 - engineering design examples, 158–166
 - exercises, 185–190
 - global optimality, 149–158
 - irregular points, 178–179
 - physical meaning of Lagrange multipliers, 143–148
 - postoptimality analysis, 143–148
 - second-ordered conditions for constrained optimization, 179–184
 - sufficiency check for rectangular beam design problem, 184–185
 - unconstrained optimum design problems, 103–119
- Concepts, optimum design (exercises), 166–174
 - constrained optimum design problems, 168–172
 - engineering design examples, 174
 - global optimality, 173–174
 - physical meaning of Lagrange multipliers, 172–173
 - review of some basic calculus concepts, 166–167
 - unconstrained optimum design problems, 167–168
- Concepts, solution, 515–516, 548–550
 - compromise solution, 550
 - efficiency and dominance, 549–550
 - Pareto optimality, 548–549
 - utopia point, 550
 - weak Pareto optimality, 549
- Condition, descent, 389–393

- Conditions
 - second-ordered, 179–184
 - transformation of KKT, 384–385
- Conditions, alternate form of KKT necessary, 175–178
 - example—alternate form of KKT conditions, 176–177
 - example—check for KKT necessary conditions, 177–178
- Conditions, concepts relating to optimality, 103–104
- Conjugate gradient method, 296–299
 - example—use of conjugate gradient algorithm, 298
 - example—use of Excel Solver, 299
- Constraint correction (CC), 493
- Constant cost, algorithm for constraint correction at, 442–445
- Constrained design, numerical methods for, 339–377, 379–412
 - algorithms and constrained problems, 340–342
 - approximate step size determination, 388–399
 - basic concepts and ideas, 340–346
 - constrained quasi-Newton methods, 400–407
 - constrained steepest descent method, 363–369
 - constraint normalization, 343–345
 - constraint status at design point, 342–343
 - convergence of algorithms, 345–346
 - descent function, 345
 - engineering design optimization using Excel Solver, 369–373
 - examples—constraint normalization and status at point, 344–345
 - exercises, 373–377, 411–412
 - linearization of constrained problem, 346–352
 - miscellaneous numerical optimization methods, 407–411
 - potential constraint strategy, 379–382
 - QP problem, 383–388
 - QP subproblem, 358–363
 - SLP algorithm, 352–358
- Constrained design, numerical methods for (exercises), 373–377, 411–412
 - approximate step size determination, 411–412
 - basic concepts and ideas, 373–374
 - constrained quasi-Newton methods, 412
 - CSD method, 377
 - engineering design optimization using Excel Solver, 377
 - linearization of constrained problem, 374
 - quadratic programming subproblem, 375–376
 - sequential linear programming algorithm, 374–375
- Constrained design problems, 119–143, 418–420
 - example—constrained minimization problem using `fmincon`, 418–420
 - example—constrained optimum point, 120
 - example—cylindrical tank design, 127
 - example—equality constrained problem, 127
 - example—`fmincon` in Optimization Toolbox, 418–420
 - example—inequality constrained problem, 128–130
 - example—infeasible problem, 121
 - example—Lagrange multipliers and their geometrical meaning, 122–125
 - example—solution of KKT necessary conditions, 134–137, 137–140
 - example—solutions of KKT necessary conditions, 132–134
 - example—unconstrained optimum point for constrained problem, 120
 - example—use of Lagrange multipliers, 127
 - example—use of necessary conditions, 128–130
 - inequality constraints, 128–140
 - KKT, 128–140
 - necessary conditions, 128–140
 - necessary conditions: equality constraints, 121–128
 - role of constraints, 119–121
 - solution of KKT necessary conditions using Excel, 140–141
 - solution of KKT necessary conditions using MATLAB, 141–143
- Constrained optimization, second-ordered conditions for, 179–184
 - example—check for sufficient conditions, 181–182, 182–183, 183–184
- Constrained problems
 - concepts related to algorithms for, 340–342
 - solution of, 332–334
- Constrained problems, linearization of, 346–352
 - example—definition of linearized subproblem, 348–350
 - example—linearization of rectangular beam design problem, 350–352
- Constrained quasi-Newton methods, 400–407
 - descent functions, 406–407
 - deviation of QP subproblem, 400–402
 - example—use of constrained quasi-Newton method, 404–405
 - modified constrained steepest descent algorithm, 404–405
 - observations on, 406

- observations on constrained quasi-Newton methods, 406
 - quasi-Newton Hessian approximation, 403–404
 - Constrained steepest descent (CSD), 368, 388, 409, 436
 - CSD algorithm, 368
 - CSD algorithm, modified, 404–405
 - CSD algorithm observations, 368–369
 - CSD method, 363–369
 - descent function, 364–365
 - example—calculation of descent function, 365, 366–367
 - example—golden section search, 366–367
 - step size determination, 366–367
 - Constrained variable metric (CVM), 400
 - Constraint correction, algorithm for, 442–445
 - Constraint correction with minimum increase in cost, 446–447
 - Constraint limits, effect of changing, 143–145
 - Constraint normalization, 343–345
 - Constraint status at design point, 342–343
 - Constraint strategy, potential, 379–382, 478
 - example—determination of potential constraint set, 380–381
 - example—search direction and potential constraint strategy, 381–382
 - Constraint tangent hyperplane, 179
 - Constraints
 - linear, 192–193
 - notation for, 9
 - role of, 119–121
 - Constraints, identification of, 17–18
 - equality and inequality constraints, 18
 - feasibility design, 18
 - implicit constraints, 18
 - linear and nonlinear constraints, 18
 - Continuous variable optimization, 492
 - Contours
 - plotting of function, 64
 - plotting of objective function, 63
 - Control effort problem, minimum, 503–505
 - Control of systems by nonlinear programming, optimal, 493–508
 - Control, optimal, 6–7
 - Control problems
 - minimum time, 505–508
 - prototype optimal, 493–497
 - Controlled random search (CRS), 575–578, 586
 - Conventional versus optimum design, 4–6
 - Convex functions, 151–153
 - Convex programming problem, 153–156, 157–158
 - Convex sets, 149–151
 - Correction algorithm, constraint, 440–442
 - Correction, algorithm for constraint, 442–445
 - Cost
 - algorithm for constraint correction at constant, 442–445
 - algorithm for constraint correction at specified increase in, 445
 - constraint correction with minimum increase in, 446–447
 - Cost coefficients, ranging, 239–241
 - Cost function, artificial, 219–220
 - Cost function scaling, effect on Lagrange multipliers, 146–147
 - Cost reduction algorithm, 436–440
 - Covering methods, 568
 - Criterion, acceptance, 578
 - Criterion method, weighted global, 556–558
 - Criterion space and design space, 546–548
 - CRS. *See* Controlled random search
 - CSD. *See* Constrained steepest descent
 - CSD algorithm with appropriate step size, 393–399
 - Curve fitting, quadratic, 306–308
 - Cylindrical tank design, minimum cost, 35
- D**
- Data preparation, interactive, 448
 - Davidon-Fletcher-Powell (DFP) method, 324–327
 - DE. *See* Domain elimination
 - Decision making, interactive, 449–450
 - Definite matrices, quadratic forms and, 96–102
 - Definitions, standard LP, 193–194
 - Degenerate basic feasible solution, 226–228
 - Descent algorithm, modified constrained steepest, 404–405
 - Descent condition, 389–393
 - Descent directions
 - and descent step, 280–281
 - orthogonality of steepest, 314–315
 - Descent functions, 345, 406–407
 - Descent method, steepest, 293–296, 310–315
 - example—verification of properties of gradient vector, 312–314
 - orthogonality of steepest descent directions, 314–315
 - properties of gradient vector, 310–314
 - Descent, methods of generalized, 569–571
 - Descent search, steepest, 660–669
 - Descent step, 280–281
 - Descents and ascents, alternation of, 570
 - Design
 - column, 421–425
 - conventional versus optimum, 4–6
 - engineering, 4
 - flywheel, 425–429

- Design (*continued*)
 - insulated spherical tank, 20–22
 - minimum cost cylindrical tank, 35
 - minimum weight tubular column, 32–35
 - optimum, 6–7
- Design concepts and methods, discrete variable, 513–530
 - basic concepts and definitions, 514–516
 - BBM, 516–521
 - dynamic rounding-off method, 524–525
 - exercises, 527–530
 - IP, 521
 - methods for linked discrete variables, 525–526
 - neighborhood search method, 525
 - SA, 522–524
 - selection of methods, 526–527
 - sequential linearization methods, 522
- Design concepts and methods, multiobjective, 543–563
 - bounded objective function method, 558–559
 - criterion space and design space, 546–548
 - example—single-objective optimization problem, 544
 - example—two-objective optimization problem, 545
 - exercises, 560–563
 - generation of Pareto optimal set, 551–552
 - goal programming, 559
 - lexicographic method, 558
 - multiobjective GA, 552–555
 - normalization of objective functions, 552
 - optimization engine, 552
 - preferences and utility functions, 551
 - problem definition, 543–546
 - scalarization methods, 551
 - selection of methods, 559–560
 - solution concepts, 548–550
 - terminology and basic concepts, 546–552
 - vector methods, 551
 - weighted global criterion method, 556–558
 - weighted min-max method, 556
 - weighted sum method, 555–556
- Design concepts, optimum, 83–174, 175–190
 - alternate form of KKT necessary conditions, 175–178
 - basic calculus concepts, 89–103
 - constrained optimum design problems, 119–143
 - engineering design examples, 158–166
 - exercises, 185–190
 - global optimality, 149–158
 - irregular points, 178–179
 - physical meaning of Lagrange multipliers, 143–148
 - postoptimality analysis, 143–148
 - second-ordered conditions for constrained optimization, 179–184
 - sufficiency check for rectangular beam design problem, 184–185
 - unconstrained optimum design problems, 103–119
- Design concepts, optimum (exercises), 166–174
 - constrained optimum design problems, 168–172
 - engineering design examples, 174
 - global optimality, 173–174
 - physical meaning of Lagrange multipliers, 172–173
 - review of some basic calculus concepts, 166–167
 - unconstrained optimum design problems, 167–168
- Design, discrete variable optimum, 491–493
 - continuous variable optimization, 492
 - discrete variable optimization, 492–493
- Design examples, engineering, 158–166
- Design examples with MATLAB, optimum, 420–429
- Design, GA for optimum, 531–542
 - applications, 539–540
 - basic concepts and definitions, 532–534
 - exercises, 540–542
 - fundamentals of GA, 534–538
 - GA for sequencing-type problems, 538–539
- Design, general mathematical model for optimum, 41–45
 - active/inactive/violated constraints, 45
 - discrete integer design variables, 44–45
 - feasibility set, 45
 - maximization problem treatment, 43
 - standard design optimization model, 42–43
 - treatment of greater than type constraints, 43–44
- Design, global optimization concepts and methods for, 565–592
 - basic concepts of solution methods, 565–567
 - deterministic methods, 567–572
 - exercises, 588–592
 - numerical performance of methods, 585–588
 - stochastic methods, 572–579
 - two local-global stochastic methods, 579–585
- Design, introduction to, 1–14
 - basic terminology and notation, 7–14
 - conventional versus optimum design process, 4–6
 - design process, 2–4
 - engineering design versus engineering analysis, 4
 - optimum design versus optimal control, 6–7

- Design, linear programming methods for
 - optimum, 191–258, 259–275
 - alternate Simplex method, 262–263
 - artificial variables, 218–228
 - basic concepts related to linear programming problems, 195–201
 - basic ideas and steps of Simplex method, 201–218
 - definition of standard linear programming problem, 192–195
 - derivation of Simplex method, 259–261
 - duality in linear programming, 263–274
 - exercises, 246–258, 275
 - postoptimality analysis, 228–243
 - solution of LP problems using Excel Solver, 243–246
 - two-phase Simplex method, 218–228
- Design, linear programming methods for optimum (exercises), 246–258
 - basic concepts related to linear programming problem, 249–250
 - basic ideas and concepts of Simplex method, 250–252
 - definition of standard linear programming problem, 246–248
 - postoptimality analysis, 256–258
 - two phase Simplex method—artificial variables, 253–256
- Design, numerical methods for constrained, 339–377, 379–412
 - algorithms and constrained problems, 340–342
 - approximate step size determination, 388–399
 - basic concepts and ideas, 340–346
 - constrained quasi-Newton methods, 400–407
 - constrained steepest descent method, 363–369
 - constraint normalization, 343–345
 - constraint status at design point, 342–343
 - convergence of algorithms, 345–346
 - descent function, 345
 - engineering design optimization using Excel Solver, 369–373
 - examples—constraint normalization and status at point, 344–345
 - linearization of constrained problem, 346–352
 - miscellaneous numerical optimization methods, 407–411
 - potential constraint strategy, 379–382
 - QP problem, 383–388
 - QP subproblem, 358–363
 - SLP algorithm, 352–358
- Design, numerical methods for constrained (exercises), 373–377, 411–412
 - approximate step size determination, 411–412
 - basic concepts and ideas, 373–374
 - constrained quasi-Newton methods, 412
 - CSD method, 377
 - engineering design optimization using Excel Solver, 377
 - linearization of constrained problem, 374
 - quadratic programming subprogram, 375–376
 - sequential linear programming algorithm, 374–375
- Design, numerical methods for unconstrained (exercises), 300–304, 335–337
 - basic ideas and algorithm for step size determination, 300–302
 - conjugate gradient method, 303
 - exercises, 300–304
 - general concepts related to numerical algorithm, 300
 - search direction determination, 302–303
 - steepest descent method, 302–303
- Design, numerical methods for unconstrained optimum, 277–304, 305–337
 - concepts related to numerical algorithms, 278–282
 - conjugate gradient method, 296–299
 - engineering applications of unconstrained methods, 329–332
 - ideas and algorithms for step size determination, 282–293
 - Newton's method, 318–324
 - quasi-Newton methods, 324–328
 - scaling of design variables, 315–318
 - search direction determination, 293–296, 296–299, 318–324, 324–328
 - solution of constrained problems, 332–334
 - steepest descent method, 293–296, 310–315
 - step size determination, 305–310
 - unconstrained optimization methods, 332–334
- Design of
 - cabinet, 30–32
 - can, 18–20
 - rectangular beam, 162–166
 - two-bar bracket, 24–29
 - wall bracket, 158–162
- Design optimization
 - applications with implicit functions, 465–511
 - engineering, 369–373
 - role of computers in interactive, 434–435
- Design optimization algorithms, interactive, 436–448
- Design optimization applications with implicit functions
 - discrete variable optimum design, 491–493
 - exercises, 508–511

- Design optimization applications with implicit functions (*continued*)
 - formulation of practical design optimization problems, 466–472
 - general-purpose software, 479–481
 - gradient evaluation for implicit functions, 473–477
 - issues in practical design optimization, 478–479
 - multiple performance requirements, 483–491
 - optimal control of systems by nonlinear programming, 493–508
 - optimum design of three-bar structure, 483–491
 - optimum design of two-member frame, 481–483
 - out-of plane loads, 481–483
- Design optimization, interactive, 433–463
 - desired interactive capabilities, 448–450
 - examples of interactive design optimization, 454–461
 - exercises, 462–463
 - interactive design optimization algorithms, 436–448
 - interactive design optimization software, 450–454
 - role of interaction in design optimization, 434–435
- Design optimization, issues in practical, 478–479
 - attributes of good optimization algorithm, 478–479
 - potential constraint strategy, 478
 - robustness, 478
 - selection of algorithm, 478
- Design optimization, role of interaction in, 434–435
 - choosing interactive design optimization, 435
 - interactive design optimization defined, 434
 - role of computers in interactive design optimization, 434–435
- Design optimization software, interactive, 450–454
- Design point, constraint status at, 342–343
- Design problem
 - graphical solution for beam, 69–72
 - sufficiency check for rectangular beam, 184–185
- Design problem formulation, optimum, 15–54
 - design of cabinet, 30–32
 - design of can, 18–20
 - design of coil springs, 36–38
 - design of two-bar bracket, 24–29
 - exercises, 46–54
 - general mathematical model for optimum design, 41–45
 - insulated spherical tank design, 20–22
 - minimum cost cylindrical tank design, 35
 - minimum weight design of symmetric three-bar truss, 38–41
 - minimum weight tubular column design, 32–35
 - problem formulation process, 16–18
 - saw mill operation, 22–23
- Design problem formulation, optimum (exercises), 46–54
 - beam of rectangular cross-section, 50
 - beer mug, 46
 - can, 48
 - cantilever beam, 53–54
 - circular tank, 48
 - cost transportation system, 49
 - crude oil refinery, 46
 - diet of bread and milk, 47
 - electric generators, 49
 - hollow circular beam, 54
 - manufacturer's products, 47
 - multistory office building, 46
 - parallel flow heat exchanger, 46–47
 - parking lot, 47
 - shipping container, 48
 - steel framework, 48
 - still in bathtub, 47
 - tonnage of iron ore, 52–53
 - top rectangular container to transport materials, 48
 - two-bar truss, 49–50
 - vegetable oil stocks, 50–51
 - water canal, 53
- Design problems
 - classification of mixed variable optimum, 514–515
 - graphical solutions for beam, 69–72
 - with multiple solutions, 66
- Design problems, constrained optimum, 119–143, 418–420
 - example—constrained minimization problem using `fmincon`, 418–420
 - example—constrained optimum point, 120
 - example—cylindrical tank design, 127
 - example—equality constrained problem, 127
 - example—`fmincon` in Optimization Toolbox, 418–420
 - example—inequality constrained problem, 128–130
 - example—infeasible problem, 121
 - example—Lagrange multipliers and their geometrical meaning, 122–125
 - example—solution of KKT necessary conditions, 134–137, 137–140

- example—solutions of KKT necessary conditions, 132–134
- example—unconstrained optimum point for constrained problem, 120
- example—use of Lagrange multipliers, 127
- example—use of necessary conditions, 128–130
- inequality constraints, 128–140
- KKT, 128–140
- necessary conditions, 128–140
- necessary conditions: equality constraints, 121–128
- role of constraints, 119–121
- solution of KKT necessary conditions using Excel, 140–141
- solution of KKT necessary conditions using MATLAB, 141–143
- Design problems, global optimization of structural, 587–588
- Design problems, unconstrained optimum, 103–119, 415–418
 - concepts relating to optimality conditions, 103–104
 - example—adding constant to function, 111
 - example—cylindrical tank design, 113–114
 - example—determination of local minimum points, 105–106
 - example—effects of scaling, 111
 - example—local minima for function of two variables, 112–113, 115–116
 - example—local minimum points using necessary conditions, 107–108
 - example—minimum cost spherical tank using necessary conditions, 108–109
 - example—multivariable unconstrained minimization, 416–418
 - example—numerical solution of necessary conditions, 114–115
 - example—single-variable unconstrained minimization, 415–416
 - example—using necessary conditions, 105–106, 113–114
 - example—using optimality conditions, 112–113, 115–116
 - optimality conditions for functions of several variables, 109–116
 - optimality conditions for functions of single variables, 104–109
 - roots of nonlinear equations using Excel, 116–119
- Design process, 2–4
- Design representation, 532
- Design space, 546–548
- Design variables, scaling of, 315–318
 - example—effect of scaling of design variables, 315–317, 317–318
- Designs, multiple optimum, 66
- Desirable direction, 280
- Determinant, 619–621
- Determination, approximate step size, 388–399
 - basic idea, 388–389
 - CSD algorithm with appropriate step size, 393–399
 - descent condition, 389–393
 - example—calculations for step size, 391–393
 - example—constrained steepest descent method, 391–393
 - example—effect of γ on performance of CSD algorithm, 396–397
 - example—minimum area design of rectangular beam, 398–399
 - example—penalty parameter R and CSD algorithm, 397–398
 - example—use of constrained steepest descent algorithm, 393–396
- Determination, search direction, 318–324, 324–328
- Determination, step size, 305–310
 - example—alternate quadratic interpolation, 308–309
 - example—one-dimensional minimization, 308–309
 - example—quadratic interpolation, 307–308
 - inaccurate line search, 309–310
 - polynomial interpolation, 306–309
- Deterministic methods, 567–572
 - covering methods, 568
 - methods of generalized descent, 569–571
 - tunneling method, 571–572
 - zooming method, 568–569
- Diagonal matrix, 616
- Diagrams, cash flow, 594
- Direct Hessian updating, 327–328
- Directions
 - descent, 280–281
 - desirable, 280
 - method of feasibility, 407–409
 - orthogonality of steepest descent, 314–315
- Discrete variable defined, 513
- Discrete variable optimization, 492–493
- Discrete variable optimum design, 491–493
 - continuous variable optimization, 492
 - discrete variable optimization, 492–493
- Discrete variable optimum design concepts and methods, 513–530
 - basic concepts and definitions, 514–516
 - BBM, 516–521
 - dynamic rounding-off method, 524–525
 - exercises, 527–530

- Discrete variable optimum design concepts and methods (*continued*)
 - IP, 521
 - methods for linked discrete variables, 525–526
 - neighborhood search method, 525
 - SA, 522–524
 - selection of methods, 526–527
 - sequential linearization methods, 522
- Discrete variables, methods for linked, 525–526
- Domain elimination (DE), 586, 587
 - method, 580–582
- Dominance, efficiency and, 549–550
- Dynamic rounding-off method, 524–525

- E
- Economic analysis, 593–609
 - economic bases for comparison, 598–604
 - exercises, 604–609
 - time value of money, 593–598
- Economic bases for comparison, 598–604
 - alternate power stations, 602
 - annual base comparisons, 599–601
 - example—alternate designs, 599, 602
 - example—alternate power stations, 600
 - example—alternate quarries, 600–601, 603
 - PW comparisons, 601
- Economic formulas, basic, 594–598
- Efficiency and dominance, 549–550
- Eigenvalues and eigenvectors, 642–643
 - example—calculation of eigenvalues and eigenvectors, 642–643
- Eigenvectors, eigenvalues and, 642–643
- Elements, off-diagonal, 616
- Elimination, Gauss-Jordan, 625–628
- Elimination method, domain, 580–582
- Engine, optimization, 552
- Engineering applications of unconstrained methods, 329–332
- Engineering design examples, 158–166
 - design of rectangular beam, 162–166
 - design of wall bracket, 158–162
- Engineering design optimization using Excel Solver, 369–373
 - data and information collection, 369–370
 - identification/definition of design variables, 370
 - identification of constraints, 370–371
 - identification of criterion to be optimized, 370
 - project/problem statement, 369
 - solution, 373
 - Solver dialog box, 372–373
 - spreadsheet layout, 371–372
- Engineering design versus engineering analysis, 4
- Equal interval search, 286–288, 657–660
 - alternate, 288–289
- Equations
 - general solution of $m \times n$ linear, 629–635
 - solution of m linear, 628–635
 - solution of nonlinear, 331–332
- Equations, multiple nonlinear, 650–654
 - example—Newton-Raphson method, 653–654
 - example—roots of nonlinear equations, 653–654
- Equations, numerical method for nonlinear, 647–655
 - exercises, 655
 - multiple nonlinear equations, 650–654
 - single nonlinear equation, 647–650
- Equations, roots of nonlinear, 116–119
- Equations, single nonlinear, 647–650
 - example—Newton-Raphson method, 649–650
 - example—roots of nonlinear equation, 649–650
- Errors, minimization of, 497–503
- Evaluation, gradient, 465
 - example—Lagrange multipliers Postoptimality analysis, 146–147
- Examples
 - \leq type constraints, 236–237
 - \leq type constraints, 239–240
 - adding constant to function, 111
 - alternate designs, 599, 602
 - alternate form of KKT conditions, 176–177
 - alternate power stations, 600, 602
 - alternate quadratic interpolation, 308–309
 - alternate quarries, 600–601
 - analytical step size determination, 284–285
 - application of BFGS method, 328
 - application of DFP method, 325–327
 - BBM with local minimizations, 519–520
 - BBM with only discrete values allowed, 517–518
 - Big-M method for equality and \geq type constraints, 262–263
 - bolt insertion sequence determination, 539
 - calculation of descent function, 365, 366–367
 - calculation of eigenvalues and eigenvectors, 642–643
 - calculation of gradient vector, 91
 - calculations for gradient of quadratic form, 102
 - calculations for Hessian of quadratic form, 102
 - calculations for step size, 391–393

canonical form and tableau, 204
 capital recovery factor, 598
 characterization of solution for LP problems, 196–198
 check for KKT conditions at irregular points, 178–179
 check for KKT necessary conditions, 177–178
 check for sufficient conditions, 181–182, 182–183, 183–184
 checking for convexity of function, 152, 152–153
 checking for convexity of problem, 154–155, 155–156, 157–158
 checking for convexity of sets, 150
 checking for descent condition, 281
 checking for linear independence of vectors, 636–637
 checking for vector spaces, 639–641
 conjugate gradient and modified Newton methods, 323
 constrained minimization problem using `fmincon`, 418–420
 constrained optimum point, 120
 constrained problem, 85–87
 constrained steepest descent method, 391–393
 constraint correction at constant cost, 443–444, 444–445
 constraint correction at specified increase in cost, 445
 constraint correction step, 441–442
 constraint correction with minimum increase in cost, 446–447
 constraint normalization and status at point, 344–345
 conversion to standard LP form, 194–195
 cost reduction step, 437–439
 cost reduction step with potential constraints, 440
 cylindrical tank design, 113–114, 127
 definition of linearized subproblem, 348–350
 definition of QP subproblem, 359–361
 design of two-member frame, 469–472
 determinant of matrix by Gaussian elimination, 624–625
 determination of form of matrix, 99, 100–101
 determination of local minimum points, 105–106
 determination of potential constraint set, 380–381
 dual of LP program, 265
 dual of LP with equality and \geq type constraints, 265–266
 effect of γ on performance of CSD algorithm, 396–397
 effect of scaling constraint, 147–148
 effect of scaling cost function, 146–147
 effect of scaling of design variables, 315–317, 317–318
 effects of scaling, 111
 equality and \geq type constraints, 237–238, 240–241
 equality constrained problem, 127
 evaluation of gradient and Hessian of function, 92–93
 existence of global minimum, 89
 feasible problem, 223–224
`fmincon` in Optimization Toolbox, 418–420
 Gauss-Jordan reduction, 626–627
 Gauss-Jordan reduction process in tabular form, 634–635
 general solution by Gauss-Jordan reduction, 631–633
 golden section search, 366–367
 gradient evaluation for two-member frame, 474–477
 graphical representation of constrained minima, 87–88
 graphical representation of maxima, 88
 identification of unbounded problem with Simplex method, 218
 implications of degenerate feasible solution, 226–228
 inequality constrained problem, 128–130
 infeasible problem, 121
 inverse of matrix by cofactors, 626–627
 Lagrange multipliers, 146–147, 147–148
 Lagrange multipliers and their geometrical meaning, 122–125
 linear Taylor's expansion of function, 95–96
 linearization of rectangular beam design problem, 350–352
 local minima for function of two variables, 112–113, 115–116
 local minimum points using necessary conditions, 107–108
 LP problem with multiple solutions, 216–217
 matrix of quadratic form, 98
 minimization of function by golden section search, 292–293
 minimization of total potential energy of two-bar truss, 330–331
 minimum area design of rectangular beam, 398–399
 minimum cost spherical tank using necessary conditions, 108–109
 multiplication of matrices, 614, 615

- Examples (*continued*)
- multivariable unconstrained minimization, 416–418
 - Newton-Raphson method, 649–650, 653–654
 - numerical solution of necessary conditions, 114–115
 - one-dimensional minimization, 308–309
 - one-dimensional minimization with quadratic interpolation, 307–308
 - optimum cost function, 145
 - optimum design, 420–429
 - optimum design of two-bar bracket, 28–29
 - penalty parameter R and CSD algorithm, 397–398
 - pivot step, 205–206
 - of practical design optimization problems, 467–472
 - primal and dual solutions, 269–271
 - profit maximization problem, 196–198
 - ranges for cost coefficients, 239–240, 240–241
 - ranges for resource limits, 236–237, 237–238
 - rank determination by elementary operation, 629
 - recovery of Lagrange multipliers for \leq type constraint, 230–232
 - recovery of Lagrange multipliers for $=$ and \geq type constraints, 232–234
 - recovery of primal formulation from dual formulation, 267
 - representation of unconstrained minimum, 85–87
 - roots of nonlinear equation, 649–650
 - roots of nonlinear equations, 653–654
 - roots of nonlinear equations by unconstrained minimization, 331–332
 - search direction and potential constraint strategy, 381–382
 - sequential linear programming algorithm, 354–355
 - single-objective optimization problem, 544
 - single payment compound amount factor, 595
 - single payment present worth factor, 595
 - single-variable unconstrained minimization, 415–416
 - sinking fund deposit factor, 597
 - solution of equations by Gaussian elimination, 623–624
 - solution of KKT necessary conditions, 134–137, 137–140
 - solution of profit maximization problem by Simplex method, 214–216
 - solution of QP problem, 386–388
 - solution of QP subproblem, 362–363
 - solutions of KKT necessary conditions, 132–134
 - steepest descent and conjugate gradient methods, 323
 - Taylor's expansion of a function of one variable, 94
 - Taylor's expansion of a function of two variables, 95
 - two-objective optimization problem, 545
 - unbounded problem, 225–226
 - unconstrained optimum point for constrained problem, 120
 - uniform series compound amount factor, 596
 - uniform series present worth factor, 598
 - use of artificial variables for \geq type constraints, 221–223
 - use of artificial variables for equality constraints, 223–224
 - use of conjugate gradient algorithm, 298
 - use of constrained quasi-Newton method, 404–405
 - use of constrained steepest descent algorithm, 393–396
 - use of Excel Solver, 299
 - use of final primal tableau to recover dual solutions, 272–273
 - use of Lagrange multipliers, 127
 - use of modified Newton's method, 320–321, 321–322
 - use of necessary conditions, 128–130
 - use of sequential linear programming, 356–357
 - use of steepest descent algorithm, 294–295, 295–296
 - using necessary conditions, 105–106, 113–114
 - using optimality conditions, 112–113, 115–116
 - using Weierstrass theorem, 89
 - variations of constraint limits, 145
 - verification of properties of gradient vector, 312–314
- Examples, engineering design, 158–166
- design of rectangular beam, 162–166
 - design of wall bracket, 158–162
- Examples of interactive design optimization, 454–461
- Excel, roots of nonlinear equations using, 116–119
- Excel Solver, engineering design using, 369–373
- data and information collection, 369–370
 - identification/definition of design variables, 370
 - identification of constraints, 370–371

- identification of criterion to be optimized, 370
- project/problem statement, 369
- solution, 373
- Solver dialog box, 372–373
- spreadsheet layout, 371–372
- Excel Solver, solution of LP problems using, 243–246
- Exercises, for
 - Chapter 2, 46
 - Chapter 3, 72
 - Chapter 4, 166
 - Chapter 5, 185
 - Chapter 6, 246
 - Chapter 7, 275
 - Chapter 8, 300
 - Chapter 9, 335
 - Chapter 10, 373
 - Chapter 11, 411
 - Chapter 12, 429
 - Chapter 13, 462
 - Chapter 14, 508
 - Chapter 15, 527
 - Chapter 16, 540
 - Chapter 17, 560
 - Chapter 18, 588
 - Appendix A, 604
 - Appendix B, 645
 - Appendix C, 655
- Expansion, Taylor's, 93–96
- Expressions, variables and, 413–414
- F
- Feasibility directions, method of, 407–409
- Feasible region, identification of, 62–63
- Feasible solution, degenerate basic, 226–228
- Filters, Pareto-set, 554
- Fitness functions, Pareto, 554
- Fitting, quadratic curve, 306–308
- Flywheel design for minimum mass, 425–429
 - data and information collection, 426–427
 - identification/definition of design variables, 427
 - identification of constraints, 427–429
 - identification of criterion to be optimized, 427
 - project/problem statement, 425–426
- Forms, canonical, 202–203
- Formulas, basic economic, 594–598
- Formulation, design problem, 15–54
 - design of cabinet, 30–32
 - design of can, 18–20
 - design of coil springs, 36–38
 - design of two-bar bracket, 24–29
 - exercises, 46–54
 - general mathematical model for optimum design, 41–45
 - insulated spherical tank design, 20–22
 - minimum cost cylindrical tank design, 35
 - minimum weight design of symmetric three-bar truss, 38–41
 - minimum weight tubular column design, 32–35
 - problem formulation process, 16–18
 - saw mill operation, 22–23
- Formulation, design problem (exercises), 46–54
 - beam of rectangular cross-section, 50
 - beer mug, 46
 - can, 48
 - cantilever beam, 53–54
 - circular tank, 48
 - cost transportation system, 49
 - crude oil refinery, 46
 - diet of bread and milk, 47
 - electric generators, 49
 - hollow circular beam, 54
 - manufacturer's products, 47
 - multistory office building, 46
 - parallel flow heat exchanger, 46–47
 - parking lot, 47
 - shipping container, 48
 - steel framework, 48
 - still in bathtub, 47
 - tonnage of iron ore, 52–53
 - top rectangular container to transport materials, 48
 - two-bar truss, 49–50
 - vegetable oil stocks, 50–51
 - water canal, 53
- Formulation process, problem, 16–18
 - data and information collection, 16
 - identification/definition of design variables, 16–17
 - identification of constraints, 17–18
 - identification of criterion to be optimized, 17
 - project/problem statement, 16
- Formulations, comparison of three, 508
- Frame, optimum design of two-member, 481–483
- Function contours
 - plotting of, 64
 - plotting of objective, 63
- Function method, bounded objective, 558–559
- Functions
 - artificial cost, 219–220
 - descent, 345, 406–407
 - normalization of objective, 552
 - Pareto fitness, 554
 - plotting, 61–62
 - utility, 551

- Functions, convex, 151–153
- Functions, design applications with implicit, 465–511
 - discrete variable optimum design, 491–493
 - exercises, 508–511
 - formulation of practical design optimization problems, 466–472
 - general-purpose software, 479–481
 - gradient evaluation for implicit functions, 473–477
 - issues in practical design optimization, 478–479
 - multiple performance requirements, 483–491
 - optimal control of systems by nonlinear programming, 493–508
 - optimum design of three-bar structure, 483–491
 - optimum design of two-member frame, 481–483
 - out-of plane loads, 481–483
- Functions, gradient evaluation for implicit, 473–477
 - example—gradient evaluation for two-member frame, 474–477
- Functions of single variables, optimality conditions for, 104–109

- G
- GA. *See* Genetic algorithms
- Gauss-Jordan elimination, 625–628
- Gaussian elimination procedure, 621–625
- Gene defined, 532
- General-purpose software, 479–481, 480–481
 - integration of application into, 480–481
 - software selection, 480
- Generalized descent, methods of, 569–571
- Generalized reduced gradient (GRG) method, 410–411
- Generation defined, 532
- Genetic algorithms (GA) for optimum design, 531–542
 - applications, 539–540
 - basic concepts and definitions, 532–534
 - exercises, 540–542
 - fundamentals of GA, 534–538
 - GA for sequencing-type problems, 538–539
- Genetic algorithms (GA) for sequencing-type problems, 538–539
 - example—bolt insertion sequence determination, 539
- Genetic algorithms (GA), fundamentals of, 534–538
 - amount of crossover and mutation, 536–537
 - crossover, 534–535
 - genetic algorithms, 537
 - immigration, 538
 - leader of population, 537
 - multiple runs for problem, 538
 - mutation, 535–536
 - reproduction procedure, 534
 - stopping criteria, 537
- Genetic algorithms (GA), multiobjective, 552–555
 - elitist strategy, 554
 - niche techniques, 555
 - Pareto fitness function, 554
 - Pareto-set filter, 554
 - ranking, 553–554
 - tournament selection, 555
 - VEGA, 553
- Global and local minima, definitions of, 84–89
- Global criterion method, weighted, 556–558
- Global optimality, 149–158
 - convex functions, 151–153
 - convex programming problem, 153–156
 - convex sets, 149–151
 - example—checking for convexity of function, 152, 152–153
 - example—checking for convexity of problem, 154–155, 155–156, 157–158
 - example—checking for convexity of sets, 150
 - sufficient conditions for convex programming problems, 157–158
 - transformation of constraint, 156–157
- Global optimization concepts and methods, 565–592
 - basic concepts of solution methods, 565–567
 - deterministic methods, 567–572
 - exercises, 588–592
 - numerical performance of methods, 585–588
 - stochastic methods, 572–579
 - two local-global stochastic methods, 579–585
- Global optimization of structural design problems, 587–588
- Goal programming, 559
- Golden section search, 289–293, 660
- Golf methods, 570–571
- Good optimization algorithm, attributes of, 478–479
- Gradient evaluation for implicit functions, 473–477
- Gradient evaluation requires special procedures, 465
- Gradient method, conjugate, 296–299
- Gradient projection method, 409–410
- Gradient vectors, 90–91
 - properties of, 310–314

- Graphical optimization, 55–82
 - design problem with multiple solutions, 66
 - exercises, 72–82
 - graphical solution for beam design problem, 69–72
 - graphical solution for minimum weight tubular column, 69
 - graphical solution process, 55–60
 - infeasible problem, 67–69
 - problem with unbounded solution, 66–67
 - use of Mathematica for graphical optimization, 60–63
 - use of MATLAB for graphical optimization, 64–66
 - Graphical optimization (exercises), 72–82
 - design of flag pole, 79–80
 - design of sign support column, 80–81
 - design of tripod, 81–82
 - design of water tower support column, 77–79
 - Graphical optimization, use of Mathematica for, 60–63
 - identification and hatching of infeasible region, 62
 - identification of feasible region, 62–63
 - identification of optimum solution, 63
 - plotting functions, 61–62
 - plotting of objective function contours, 63
 - Graphical optimization, use of MATLAB for, 64–66
 - editing graphs, 64–66
 - plotting of function contours, 64
 - Graphical solution for beam design problem, 69–72
 - Graphical solution for minimum weight tubular column, 69
 - Graphical solution procedure, step-by-step, 56–60
 - coordinate system set-up, 56–60
 - identification of feasible region, 57–58
 - identification of feasible region for inequality, 57
 - identification of optimum solution, 58–60
 - inequality constraint boundary plot, 57
 - plotting objective function contours, 58
 - Graphical solution process, 55–60
 - profit maximization problem, 55–56
 - step-by-step, 56–60
 - Graphics, interactive, 450, 457–461
 - Graphs, editing, 64–66
- H**
- Hessian approximation, quasi-Newton, 403–404
 - Hessian matrix, 92–93
 - Hessian updating
 - direct, 327–328
 - inverse, 324–327
 - Hyperplane, constraint tangent, 179
- I**
- Identity matrix, 616
 - IDESIGN
 - capabilities of, 453–454
 - user interface for, 451–452
 - Implicit functions, design applications with, 465–511
 - discrete variable optimum design, 491–493
 - exercises, 508–511
 - formulation of practical design optimization problems, 466–472
 - general-purpose software, 479–481
 - gradient evaluation for implicit functions, 473–477
 - issues in practical design optimization, 478–479
 - multiple performance requirements, 483–491
 - optimal control of systems by nonlinear programming, 493–508
 - optimum design of three-bar structure, 483–491
 - optimum design of two-member frame, 481–483
 - out-of plane loads, 481–483
 - Implicit functions, gradient evaluation for, 473–477
 - example—gradient evaluation for two-member frame, 474–477
 - Inaccurate line search, 309–310
 - Inequality, identification and hatching of infeasible region for, 62
 - Infeasible problem, 67–69
 - Infeasible region, identification and hatching of, 62
 - Information, second-order, 179
 - Insulated spherical tank design, 20–22
 - Integer programming (IP), 521
 - Integer variable defined, 513
 - Integration, stochastic, 579
 - Interaction, role in design optimization, 434–435
 - choosing interactive design optimization, 435
 - interactive design optimization defined, 434
 - role of computers in interactive design optimization, 434–435
 - Interactive algorithms, observations on, 447–448
 - Interactive capabilities, 448–449
 - Interactive capabilities, desired, 448–450
 - interactive capabilities, 448–449
 - interactive data preparation, 448

- Interactive capabilities, desired (*continued*)
 - interactive decision making, 449–450
 - interactive graphics, 450
 - Interactive data preparation, 448
 - Interactive decision making, 449–450
 - Interactive design optimization, 433–463
 - desired interactive capabilities, 448–450
 - examples of interactive design optimization, 454–461
 - exercises, 462–463
 - interactive design optimization algorithms, 436–448
 - interactive design optimization software, 450–454
 - role of computers in, 434–435
 - role of interaction in design optimization, 434–435
 - Interactive design optimization algorithms, 436–448
 - algorithm for constraint correction at constant cost, 442–445
 - algorithm for constraint correction at specified increase in cost, 445
 - constraint correction algorithm, 440–442
 - constraint correction with minimum increase in cost, 446–447
 - cost reduction algorithm, 436–440
 - example—constraint correction at constant cost, 443–444, 444–445
 - example—constraint correction at specified increase in cost, 445
 - example—constraint correction step, 441–442
 - example—constraint correction with minimum increase in cost, 446–447
 - example—cost reduction step, 437–439
 - example—cost reduction step with potential constraints, 440
 - observations on interactive algorithms, 447–448
 - Interactive design optimization, examples of, 454–461
 - formulation of spring design problem, 454–455
 - interactive graphics, 457–461
 - interactive solution for spring design problem, 455–457
 - optimum solution for spring design problem, 455
 - Interactive design optimization software, 450–454
 - capabilities of IDESIGN, 453–454
 - user interface for IDESIGN, 451–452
 - Interactive graphics, 450, 457–461
 - Interface, user, 451–452
 - Interpolation, alternate quadratic, 308–309
 - Interpolation, polynomial, 306–309
 - alternate quadratic interpolation, 308–309
 - quadratic curve fitting, 306–308
 - Interval reducing methods, 286
 - Interval search
 - alternate equal, 288–289
 - equal, 286–288, 657–660
 - Inverse Hessian updating, 324–327
 - IP. *See* Integer programming
 - Irregular points, 178–179
 - example—check for KKT conditions at irregular points, 178–179
- K**
- Karush-Kuhn-Tucker (KKT), 175, 566
 - KKT conditions, transformation of, 384–385
 - KKT. *See* Karush-Kuhn-Tucker
 - KKT necessary conditions, alternate form of, 175–178
 - example—alternate form of KKT conditions, 176–177
 - example—check for KKT necessary conditions, 177–178
 - KKT necessary conditions for QP problem, 384
- L**
- Lagrange multipliers, effect of cost function scaling on, 146–147
 - Lagrange multipliers, physical meaning of, 143–148
 - constraint variation sensitivity result, 148
 - effect of scaling constraint on Lagrange multiplier, 147–148
 - example—effect of scaling constraint, 147–148
 - example—effect of scaling cost function, 146–147
 - example—Lagrange multipliers, 146–147, 147–148
 - example—optimum cost function, 145
 - example—variations of constraint limits, 145
 - scaling cost function on Lagrange multipliers, 146–147
 - Lagrangian methods, augmented, 334
 - Length of vectors. *See* Norm/length of vectors
 - Lexicographic method, 558
 - Line search, inaccurate, 309–310
 - Linear constraints, 192–193
 - Linear equations, general solution of $m \times n$, 629–635
 - Linear equations in n unknowns, solution of n , 618–628
 - determinants, 619–621

- example—determinant of matrix by Gaussian elimination, 624–625
- example—Gauss-Jordan reduction, 626–627
- example—Gauss-Jordan reduction process in tabular form, 634–635
- example—general solution by Gauss-Jordan reduction, 631–633
- example—inverse of matrix by cofactors, 626–627
- example—rank determination by elementary operation, 629
- example—solution of equations by Gaussian elimination, 623–624
- Gauss-Jordan elimination, 625–628
- Gaussian elimination procedure, 621–625
- general solution of $m \times n$ linear equations, 629–635
- inverse of matrix, 625–628
- linear systems, 618–619
- rank of matrix, 628–629
- Linear equations, solution of n , 628–635
- Linear programming, duality in, 263–274
 - alternate treatment of equality constraints, 266–267
 - determination of primal solution from dual solution, 267–271
 - dual LP program, 264–265
 - dual variables as Lagrange multipliers, 273–274
 - example—dual of LP program, 265
 - example—dual of LP with equality and \geq type constraints, 265–266
 - example—primal and dual solutions, 269–271
 - example—recovery of primal formulation from dual formulation, 267
 - example—use of final primal tableau to recover dual solutions, 272–273
 - standard primal LP, 263–264
 - treatment of equality constraints, 265–266
 - use of dual tableau to recover primal solution, 271–273
- Linear programming methods for optimum design, 191–258, 259–275
 - alternate Simplex method, 262–263
 - artificial variables, 218–228
 - basic concepts related to linear programming problems, 195–201
 - basic ideas and steps of Simplex method, 201–218
 - definition of standard linear programming problem, 192–195
 - derivation of Simplex method, 259–261
 - duality in linear programming, 263–274
 - exercises, 246–258, 275
 - postoptimality analysis, 228–243
 - solution of LP problems using Excel Solver, 243–246
 - two-phase Simplex method, 218–228
- Linear programming methods for optimum design (exercises), 246–258, 275
 - basic concepts related to linear programming problem, 249–250
 - basic ideas and concepts of Simplex method, 250–252
 - definition of standard linear programming problem, 246–248
 - postoptimality analysis, 256–258
 - two phase Simplex method—artificial variables, 253–256
- Linear programming problem, 56
- Linear programming problem, definition of standard, 192–195
 - example—conversion to standard LP form, 194–195
 - linear constraints, 192–193
 - standard LP definition, 193–194
 - unrestricted variables, 193
- Linear programming problems, concepts related to, 191, 195–201
 - basic concepts, 195–196
 - example—characterization of solution for LP problems, 196–198
 - example—determination of basic solutions, 199–200
 - example—profit maximization problem, 196–198
 - LP terminology, 198–200
 - optimum solution for LP problems, 201
- Linear programs (LP), 191
- Linear systems, 618–619
- Linearization methods, sequential, 522
- Linearization of constrained problem, 346–352
 - example—definition of linearized subproblem, 348–350
 - example—linearization of rectangular beam design problem, 350–352
- Linked discrete variable, 513
- Linked discrete variables, methods for, 525–526
- Loads, out-of plane, 481–483
- Local-global algorithm, conceptual, 579–580
- Local minima, definitions of global and, 84–89
- Lower triangle matrix, 616
- LP definition, standard, 193–194
- LP problems
 - optimum solutions for, 201
 - solution of, 243–246
- LP terminology, 198–200

- M**
- Marquardt modification, 323–324
 - Mass
 - column design for minimum, 421–425
 - flywheel design for minimum, 425–429
 - Mathematica, use of for graphical optimization, 60–63
 - identification and hatching of infeasible region, 62
 - identification of feasible region, 62–63
 - identification of optimum solution, 63
 - plotting functions, 61–62
 - plotting of objective function contours, 63
 - Mathematical model for optimum design
 - active/inactive/violated constraints, 45
 - discrete integer design variables, 44–45
 - feasibility set, 45
 - general, 41–45
 - maximization problem treatment, 43
 - standard design optimization model, 42–43
 - treatment of greater than type constraints, 43–44
 - MATLAB, optimum design examples with, 420–429
 - column design for minimum mass, 421–425
 - flywheel design for minimum mass, 425–429
 - location of maximum shear stress, 420–421
 - two spherical bodies in contact, 420–421
 - MATLAB, optimum design with, 413–432
 - constrained optimum design problems, 418–420
 - exercises, 429–432
 - Optimization Toolbox, 413–415
 - optimum design examples with MATLAB, 420–429
 - unconstrained optimum design problems, 415–418
 - MATLAB, use of for graphical optimization, 64–66
 - editing graphs, 64–66
 - plotting of function contours, 64
 - Matrices
 - addition of, 613
 - column, 613
 - condition numbers of, 644
 - definition of, 611–613
 - diagonal, 616
 - equivalence of, 616
 - identity, 616
 - inverse of, 625–628
 - lower triangle, 616
 - multiplication of, 613–615
 - null, 613
 - partitioning of, 617–618
 - quadratic forms and definite, 96–102
 - rank of, 628–629
 - row, 613
 - scalar, 616
 - square, 616–617
 - upper triangle, 616
 - Matrices, norms and condition numbers of, 643–644
 - condition number of matrix, 644
 - norm of vectors and matrices, 643–644
 - Matrices, type of, 613–618
 - addition of matrices, 613
 - elementary row—column operations, 616
 - equivalence of matrices, 616
 - example—multiplication of matrices, 614, 615
 - multiplication of matrices, 613–615
 - null matrix, 613
 - partitioning of matrices, 617–618
 - scalar product—dot product of vectors, 616
 - square matrices, 616–617
 - vectors, 613
 - Matrix algebra, vector and, 611–646
 - concepts related to set of vectors, 635–641
 - definition of matrices, 611–613
 - eigenvalues and eigenvectors, 642–643
 - exercises, 645–646
 - norm and condition number of matrix, 643–644
 - solution of m linear equations in n unknowns, 628–635
 - type of matrices and their operations, 613–618
 - Matrix, changes in coefficient, 241–243
 - Matrix, Hessian, 92–93
 - Matrix operation, 414
 - Methods
 - A-R, 586
 - alternate Simplex, 262–263
 - augmented Lagrangian, 334
 - basic steps of Simplex, 206–211
 - BFGS, 327–328
 - bounded objective function, 558–559
 - DFP, 324–327
 - domain elimination, 580–582
 - dynamic rounding-off, 524–525
 - of generalized descent, 569–571
 - golf, 570–571
 - gradient projection, 409–410
 - GRG, 410–411
 - interval reducing, 286
 - lexicographic, 558
 - linear programming, 191–258
 - modified Newton's, 669–673
 - multiplier, 334
 - multistart, 573

- neighborhood search, 525
- observations on constrained quasi-Newton, 406
- operations analysis of, 583–585
- performance, 586
- performance of stochastic zooming, 586–587
- scalarization, 551
- sequential linearization, 522
- stochastic zooming, 582–583
- tunneling, 571–572
- two-phase Simplex, 218–228
- vector, 551
- weighted global criterion, 556–558
- weighted min-max, 556
- weighted sum, 555–556
- zooming, 568–569
- Methods, A-R (acceptance-rejection), 578
- Methods, basic concepts of solution, 565–567
- Methods, basic ideas and steps of Simplex, 201–218
 - basic steps of Simplex method, 206–211
 - canonical form/general solution of $Ax = b$, 202–203
 - example—canonical form and tableau, 204
 - example—identification of unbounded problem with Simplex method, 218
 - example—LP problem with multiple solutions, 216–217
 - example—pivot step, 205–206
 - example—solution by Simplex method, 213–214
 - example—solution of profit maximization problem, 214–216
 - interchange of basic and nonbasic variables, 205–206
 - pivot step, 205–206
 - Simplex, 202
 - Simplex algorithms, 211–212
 - steps of Simplex method, 207–211
 - tableau, 203–204
- Methods, clustering, 573–575
- Methods, conjugate gradient, 296–299
 - example—use of conjugate gradient algorithm, 298
 - example—use of Excel Solver, 299
- Methods, constrained quasi-Newton, 400–407
 - descent functions, 406–407
 - deviation of QP subproblem, 400–402
 - example—use of constrained quasi-Newton method, 404–405
 - modified constrained steepest descent algorithm, 404–405
 - observations on constrained quasi-Newton methods, 406
 - quasi-Newton Hessian approximation, 403–404
- Methods, constrained steepest descent, 363–369
 - CSD algorithm, 368
 - CSD algorithm observations, 368–369
 - descent function, 364–365
 - example—calculation of descent function, 365, 366–367
 - example of—golden section search, 366–367
 - step size determination, 366–367
- Methods, covering, 568
- Methods, deterministic, 567–572
 - covering methods, 568
 - methods of generalized descent, 569–571
 - tunneling method, 571–572
 - zooming method, 568–569
- Methods, engineering applications of
 - unconstrained, 329–332
 - example—minimization of total potential energy of two-bar truss, 330–331
 - example—roots of nonlinear equations, 331–332
 - example—unconstrained minimization, 331–332
 - minimization of total potential energy, 329–331
 - solution of nonlinear equations, 331–332
- Methods for linked discrete variables, 525–526
- Methods for optimum design, global concepts and, 565–592
 - basic concepts of solution methods, 565–567
 - deterministic methods, 567–572
 - exercises, 588–592
 - numerical performance of methods, 585–588
 - stochastic methods, 572–579
 - two local-global stochastic methods, 579–585
- Methods, miscellaneous numerical optimization, 407–411
 - gradient projection method, 409–410
 - GRG method, 410–411
 - method of feasibility directions, 407–409
- Methods, multiobjective optimum design
 - concepts and, 543–563
 - bounded objective function method, 558–559
 - criterion space and design space, 546–548
 - example—single-objective optimization problem, 544
 - example—two-objective optimization problem, 545
 - exercises, 560–563
 - generation of Pareto optimal set, 551–552
 - goal programming, 559
 - lexicographic method, 558
 - multiobjective GA, 552–555
 - normalization of objective functions, 552

- Methods, multiobjective optimum design
 - concepts and (*continued*)
 - optimization engine, 552
 - preferences and utility functions, 551
 - problem definition, 543–546
 - scalarization methods, 551
 - selection of methods, 559–560
 - solution concepts, 548–550
 - terminology and basic concepts, 546–552
 - vector methods, 551
 - weighted global criterion method, 556–558
 - weighted min-max method, 556
 - weighted sum method, 555–556
- Methods, Newton's, 318–324
 - classical Newton's method, 318–319
 - example—conjugate gradient and modified Newton methods, 323
 - example—steepest descent and conjugate gradient methods, 323
 - example—use of modified Newton's method, 320–321, 321–322
 - Marquardt modification, 323–324
 - modified Newton's method, 319–323
- Methods, numerical performance of, 585–588
 - DE methods, 586–587
 - global optimization of structural design problems, 587–588
 - performance methods using unconstrained problems, 586
 - performance of stochastic zooming method, 586–587
 - summary of features of methods, 585–586
- Methods, quasi-Newton, 324–328
 - BFGS method, 327–328
 - DFP method, 324–327
 - direct Hessian updating, 327–328
 - example—application of BFGS method, 328
 - example—application of DFP method, 325–327
 - inverse Hessian updating, 324–327
- Methods, steepest descent, 293–296, 310–315
 - example—use of steepest descent algorithm, 294–295, 295–296
 - example—verification of properties of gradient vector, 312–314
 - orthogonality of steepest descent directions, 314–315
 - properties of gradient vector, 310–314
- Methods, stochastic, 572–579
 - A-R methods, 578
 - clustering methods, 573–575
 - CRS method, 575–578
 - multistart method, 573
 - pure random search, 573
 - stochastic integration, 579
- Methods, two local-global stochastic, 579–585
 - conceptual local-global algorithm, 579–580
 - domain elimination method, 580–582
 - operations analysis of methods, 583–585
 - stochastic zooming method, 582–583
- Methods, unconstrained optimization, 332–334
 - augmented Lagrangian methods, 334
 - multiplier methods, 334
 - sequential unconstrained minimization techniques, 333–334
- Mill, saw, 22–23
- Min-max method, weighted, 556
- Minima, definitions of global and local, 84–89
 - example—constrained problem, 85–87
 - example—existence of global minimum, 89
 - example—graphical representation of constrained minima, 87–88
 - example—graphical representation of maxima, 88
 - example—graphical representation of unconstrained minimum, 85–87
 - example—using Weierstrass theorem, 89
 - existence of minimum, 89
- Minimization techniques, sequential unconstrained, 333–334
- Minimum control effort problem, 503–505
- Minimum, existence of, 89
- Minimum mass
 - column design for, 421–425
 - flywheel design for, 425–429
- Minimum weight tubular column, graphical solution for, 69
- Mixed variable optimum design problems (MV-OPT), classification of, 514–515
- Modifications, Marquardt, 323–324
- Modified constrained steepest descent algorithm, 404–405
- Money, time value of, 593–598
 - basic economic formulas, 594–598
 - cash flow diagrams, 594
 - example—capital recovery factor, 598
 - example—single payment compound amount factor, 595
 - example—single payment present worth factor, 595
 - example—sinking fund deposit factor, 597
 - example—uniform series compound amount factor, 596
 - example—uniform series present worth factor, 598
- Motion, optimal control of system, 508

- Multiobjective design concepts and methods, 543–563
 - bounded objective function method, 558–559
 - criterion space and design space, 546–548
 - example—single-objective optimization problem, 544
 - example—two-objective optimization problem, 545
 - exercises, 560–563
 - generation of Pareto optimal set, 551–552
 - goal programming, 559
 - lexicographic method, 558
 - multiobjective GA, 552–555
 - normalization of objective functions, 552
 - optimization engine, 552
 - preferences and utility functions, 551
 - problem definition, 543–546
 - scalarization methods, 551
 - selection of methods, 559–560
 - solution concepts, 548–550
 - terminology and basic concepts, 546–552
 - vector methods, 551
 - weighted global criterion method, 556–558
 - weighted min-max method, 556
 - weighted sum method, 555–556
- Multiobjective GA, 552–555
 - elitist strategy, 554
 - niche techniques, 555
 - Pareto fitness function, 554
 - Pareto-set filter, 554
 - ranking, 553–554
 - tournament selection, 555
 - VEGA, 553
- Multiple nonlinear equations, 650–654
- Multiple optimum designs, 66
- Multiple performance requirements, 483–491
 - asymmetric three-bar structure, 484–490
 - comparison of solutions, 490–491
 - symmetric three-bar structure, 483–484
- Multiple solutions, design problem with, 66
- Multiplier methods, 334
- Multipliers, physical meaning of Lagrange, 143–148
 - constraint variation sensitivity result, 148
 - effect of scaling constraint on Lagrange multiplier, 147–148
 - example—effect of scaling constraint, 147–148
 - example—effect of scaling cost function, 146–147
 - example—Lagrange multipliers, 146–147, 147–148
 - example—optimum cost function, 145
 - example—variations of constraint limits, 145
 - scaling cost function on Lagrange multipliers, 146–147
- Multistart method, 573
- N
- Necessary conditions, 128–140
- Neighborhood search method, 525
- Newton’s methods. *See also* Quasi-Newton methods, 318–324
 - classical Newton’s methods, 318–319
 - example—conjugate gradient and modified Newton methods, 323
 - example—steepest descent and conjugate gradient methods, 323
 - example—use of modified Newton’s method, 320–321, 321–322
 - Marquardt modification, 323–324
 - modified, 669–673
 - modified Newton’s method, 319–323
- Niche techniques, 555
- Nonlinear equation, single, 647–650
 - example—Newton-Raphson method, 649–650
 - example—roots of nonlinear equation, 649–650
- Nonlinear equations, multiple, 650–654
 - example—Newton-Raphson method, 653–654
 - example—roots of nonlinear equations, 653–654
- Nonlinear equations, numerical method for, 647–655
 - exercises, 655
 - multiple nonlinear equations, 650–654
 - single nonlinear equation, 647–650
- Nonlinear equations, solution of, 331–332
- Nonlinear equations using Excel, roots of, 116–119
- Nonlinear optimization problems, numerical methods for, 277
- Nonlinear programming (NLP), 277
- Nonlinear programming, control of systems by, 493–508
 - comparison of three formulations, 508
 - minimization of errors in state variables, 497–503
 - minimum control effort problem, 503–505
 - minimum time control problem, 505–508
 - optimal control of system motion, 508
 - prototype optimal control problem, 493–497
- Norm/length of vectors, 11
- Normalization, constraint, 343–345
- Notation
 - basic terminology and, 7–14
 - summation, 9–10

- Notation for constraints, 9
- Null matrix, 613
- Numerical algorithms, concepts related to, 278–282
 - convergence of algorithms, 282
 - descent direction and descent step, 280–281
 - example—checking for descent condition, 281
 - general algorithm, 279–280
 - rate of convergence, 282
- Numerical method for nonlinear equations, 647–655
 - exercises, 655
 - multiple nonlinear equations, 650–654
 - single nonlinear equation, 647–650
- Numerical methods for constrained design, 339–377, 379–412
 - approximate step size determination, 388–399
 - constrained quasi-Newton methods, 400–407
 - miscellaneous numerical optimization methods, 407–411
 - potential constraint strategy, 379–382
 - QP problem, 383–388
- Numerical methods for constrained design (exercises), 411–412
 - approximate step size determination, 411–412
 - constrained quasi-Newton methods, 412
- Numerical methods for nonlinear optimization problems, 277
- Numerical methods for unconstrained design, 277–304, 305–337
 - concepts related to numerical algorithms, 278–282
 - conjugate gradient method, 296–299
 - engineering applications of unconstrained methods, 329–332
 - exercises, 335–337
 - ideas and algorithms for step size determination, 282–293
 - Newton’s method, 318–324
 - quasi-Newton methods, 324–328
 - scaling of design variables, 315–318
 - search direction determination, 293–296, 296–299, 318–324, 324–328
 - solution of constrained problems, 332–334
 - steepest descent method, 293–296, 310–315
 - step size determination, 305–310
 - unconstrained optimization methods, 332–334
- Numerical methods for unconstrained design (exercise), 300–304
 - basic ideas and algorithm for step size determination, 300–302
 - conjugate gradient method, 303
 - general concepts related to numerical algorithm, 300
 - search direction determination, 302–303
 - steepest descent method, 302–303
- Numerical optimization methods, miscellaneous, 407–411
 - gradient projection method, 409–410
 - GRG method, 410–411
 - method of feasibility directions, 407–409
- Numerical performance of methods, 585–588
 - DE methods, 586–587
 - global optimization of structural design problems, 587–588
 - performance methods using unconstrained problems, 586
 - performance of stochastic zooming method, 586–587
 - summary of features of methods, 585–586
- O
- Objective function contours, plotting of, 63
- Objective functions, normalization of, 552
- Off-diagonal elements, 616
- Operations analysis of methods, 583–585
- Optimal control of system motion, 508
- Optimal control, optimum design versus, 6–7
- Optimal control problem, prototype, 493–497
- Optimal set, generation of Pareto, 551–552
- Optimality conditions
 - concepts relating to, 103–104
 - for functions of single variables, 104–109
- Optimality, global, 149–158
 - convex functions, 151–153
 - convex programming problem, 153–156
 - convex sets, 149–151
 - example—checking for convexity of function, 152, 152–153
 - example—checking for convexity of problem, 154–155, 155–156, 157–158
 - example—checking for convexity of sets, 150
 - sufficient conditions for convex programming problems, 157–158
 - transformation of constraint, 156–157
- Optimality, Pareto, 548–549
- Optimality weak Pareto, 549
- Optimization
 - continuous variable, 492
 - discrete variable, 492–493
 - engineering design, 369–373
 - engines, 552
 - role of computers in interactive design, 434–435
- Optimization algorithm, attributes of good, 478–479

- Optimization algorithms, interactive design, 436–448
 - algorithm for constraint correction at constant cost, 442–445
 - algorithm for constraint correction at specified increase in cost, 445
 - constraint correction algorithm, 440–442
 - constraint correction with minimum increase in cost, 446–447
 - cost reduction algorithm, 436–440
 - example—constraint correction at constant cost, 443–444, 444–445
 - example—constraint correction at specified increase in cost, 445
 - example—constraint correction step, 441–442
 - example—constraint correction with minimum increase in cost, 446–447
 - example—cost reduction step, 437–439
 - example—cost reduction step with potential constraints, 440
 - observations on interactive algorithms, 447–448
- Optimization applications with implicit functions, design, 465–511
- Optimization, examples of interactive design, 454–461
 - formulation of spring design problem, 454–455
 - interactive graphics, 457–461
 - interactive solution for spring design problem, 455–457
 - optimum solution for spring design problem, 455
- Optimization, graphical, 55–82
 - design problem with multiple solutions, 66
 - exercises, 72–82
 - graphical solution for beam design problem, 69–72
 - graphical solution for minimum weight tubular column, 69
 - graphical solution process, 55–60
 - infeasible problem, 67–69
 - problem with unbounded solution, 66–67
 - use of Mathematica for graphical optimization, 60–63
 - use of MATLAB for graphical optimization, 64–66
- Optimization, graphical, (exercises), 72–82
 - design of flag pole, 79–80
 - design of sign support column, 80–81
 - design of tripod, 81–82
 - design of water tower support column, 77–79
- Optimization, interactive design, 433–463
 - desired interactive capabilities, 448–450
 - examples of interactive design optimization, 454–461
 - exercises, 462–463
 - interactive design optimization algorithms, 436–448
 - interactive design optimization software, 450–454
 - role of interaction in design optimization, 434–435
- Optimization, issues in practical design, 478–479
 - attributes of good optimization algorithm, 478–479
 - potential constraint strategy, 478
 - robustness, 478
 - selection of algorithm, 478
- Optimization methods, miscellaneous numerical, 407–411
 - gradient projection method, 409–410
 - GRG method, 410–411
 - method of feasibility directions, 407–409
- Optimization methods, unconstrained, 332–334
 - augmented Lagrangian methods, 334
 - multiplier methods, 334
 - sequential unconstrained minimization techniques, 333–334
- Optimization problems, numerical methods for nonlinear, 277
- Optimization problems, practical design, 466–472
- Optimization, role of interaction in design, 434–435
 - choosing interactive design optimization, 435
 - interactive design optimization defined, 434
 - role of computers in interactive design optimization, 434–435
- Optimization, second-order conditions for constrained, 179–184
 - example—check for sufficient conditions, 181–182, 182–183, 183–184
- Optimization software, interactive design, 450–454
 - capabilities of IDESIGN, 453–454
 - user interface for IDESIGN, 451–452
- Optimization Toolbox, 413–415
 - array operation, 414
 - matrix operation, 414
 - scalar operation, 414
 - variables and expressions, 413–414
- Optimization, use of Mathematica for graphical, 60–63
 - identification and hatching of infeasible region for inequality, 62
 - identification of feasible region, 62–63
 - identification of optimum solution, 63

- Optimization, use of Mathematica for graphical
 - (*continued*)
 - plotting functions, 61–62
 - plotting of objective function contours, 63
- Optimization, use of MATLAB for graphical, 64–66
 - editing graphs, 64–66
 - plotting of function contours, 64
- Optimum design
 - conventional versus, 4–6
 - general mathematical model for, 41–45
- Optimum design concepts, 83–174, 175–190
 - alternate form of KKT necessary conditions, 175–178
 - basic calculus concepts, 89–103
 - constrained optimum design problems, 119–143
 - engineering design examples, 158–166
 - exercises, 185–190
 - global optimality, 149–158
 - irregular points, 178–179
 - physical meaning of Lagrange multipliers, 143–148
 - postoptimality analysis, 143–148
 - second-order conditions for constrained optimization, 179–184
 - sufficiency check for rectangular beam design problem, 184–185
 - unconstrained optimum design problems, 103–119
- Optimum design concepts and methods, discrete variable, 513–530
 - basic concepts and definitions, 514–516
 - BBM, 516–521
 - dynamic rounding-off method, 524–525
 - exercises, 527–530
 - IP, 521
 - methods for linked discrete variables, 525–526
 - neighborhood search method, 525
 - SA, 522–524
 - selection of methods, 526–527
 - sequential linearization methods, 522
- Optimum design concepts and methods,
 - multiobjective, 543–563
 - bounded objective function method, 558–559
 - criterion space and design space, 546–548
 - example—single-objective optimization problem, 544
 - example—two-objective optimization problem, 545
 - exercises, 560–563
 - generation of Pareto optimal set, 551–552
 - goal programming, 559
 - lexicographic method, 558
 - multiobjective GA, 552–555
 - normalization of objective functions, 552
 - optimization engine, 552
 - preferences and utility functions, 551
 - problem definition, 543–546
 - scalarization methods, 551
 - selection of methods, 559–560
 - solution concepts, 548–550
 - terminology and basic concepts, 546–552
 - vector methods, 551
 - weighted global criterion method, 556–558
 - weighted min-max method, 556
 - weighted sum method, 555–556
- Optimum design concepts (exercises), 166–174
 - constrained optimum design problems, 168–172
 - engineering design examples, 174
 - global optimality, 173–174
 - physical meaning of Lagrange multipliers, 172–173
 - review of some basic calculus concepts, 166–167
 - unconstrained optimum design problems, 167–168
- Optimum design, discrete variable, 491–493
 - continuous variable optimization, 492
 - discrete variable optimization, 492–493
- Optimum design examples with MATLAB, 420–429
 - column design for minimum mass, 421–425
 - flywheel design for minimum mass, 425–429
 - location of maximum shear stress, 420–421
 - two spherical bodies in contact, 420–421
- Optimum design, GA for, 531–542
 - applications, 539–540
 - basic concepts and definitions, 532–534
 - exercises, 540–542
 - fundamentals of GA, 534–538
 - GA for sequencing-type problems, 538–539
- Optimum design, global concepts and methods for, 565–592
 - basic concepts of solution methods, 565–567
 - deterministic methods, 567–572
 - exercises, 588–592
 - numerical performance of methods, 585–588
 - stochastic methods, 572–579
 - two local-global stochastic methods, 579–585
- Optimum design, linear programming methods for, 191–258, 259–275
 - alternate Simplex method, 262–263
 - artificial variables, 218–228
 - basic concepts related to linear programming problems, 195–201
 - basic ideas and steps of Simplex method, 201–218

- definition of standard linear programming problem, 192–195
- derivation of Simplex method, 259–261
- duality in linear programming, 263–274
- exercises, 246–258, 275
- postoptimality analysis, 228–243
- solution of LP problems using Excel Solver, 243–246
- two-phase Simplex method, 218–228
- Optimum design, linear programming methods for (exercises), 246–258, 275
 - basic concepts related to linear programming problem, 249–250
 - basic ideas and concepts of Simplex method, 250–252
 - definition of standard linear programming problem, 246–248
 - postoptimality analysis, 256–258
 - two phase Simplex method—artificial variables, 253–256
- Optimum design, numerical methods for constrained, 339–377, 379–412
 - algorithms and constrained problems, 340–342
 - approximate step size determination, 388–399
 - basic concepts and ideas, 340–346
 - constrained quasi-Newton methods, 400–407
 - constrained steepest descent method, 363–369
 - constraint normalization, 343–345
 - constraint status at design point, 342–343
 - convergence of algorithms, 345–346
 - descent function, 345
 - engineering design optimization using Excel Solver, 369–373
 - examples—constraint normalization and status at point, 344–345
 - exercises, 373–377
 - linearization of constrained problem, 346–352
 - miscellaneous numerical optimization methods, 407–411
 - potential constraint strategy, 379–382
 - QP problem, 383–388
 - QP subproblem, 358–363
 - SLP algorithm, 352–358
- Optimum design, numerical methods for constrained (exercises), 373–377
 - basic concepts and ideas, 373–374
 - CSD method, 377
 - engineering design optimization using Excel Solver, 377
 - linearization of constrained problem, 374
 - quadratic programming subprogram, 375–376
- sequential linear programming algorithm, 374–375
- Optimum design, numerical methods for unconstrained, 277–304, 305–337
 - concepts related to numerical algorithms, 278–282
 - conjugate gradient method, 296–299
 - engineering applications of unconstrained methods, 329–332
 - exercises, 300–304, 335–337
 - ideas and algorithms for step size determination, 282–293
 - Newton’s method, 318–324
 - quasi-Newton methods, 324–328
 - scaling of design variables, 315–318
 - search direction determination, 293–296, 296–299, 318–324, 324–328
 - solution of constrained problems, 332–334
 - steepest descent method, 293–296, 310–315
 - step size determination, 305–310
 - unconstrained optimization methods, 332–334
- Optimum design, numerical methods for unconstrained (exercises), 300–304, 335–337
 - basic ideas and algorithm for step size determination, 300–302
 - conjugate gradient method, 303
 - general concepts related to numerical algorithm, 300
 - search direction determination, 302–303
 - steepest descent method, 302–303
- Optimum design of three-bar structure, 483–491
- Optimum design of two-member frame, 481–483
- Optimum design problem formulation, 15–54
 - design of cabinet, 30–32
 - design of can, 18–20
 - design of coil springs, 36–38
 - design of two-bar bracket, 24–29
 - general mathematical model for optimum design, 41–45
 - insulated spherical tank design, 20–22
 - minimum cost cylindrical tank design, 35
 - minimum weight design of symmetric three-bar truss, 38–41
 - minimum weight tubular column design, 32–35
 - problem formulation process, 16–18
 - saw mill operation, 22–23
- Optimum design problem formulation (exercises), 46–54
 - beam of rectangular cross-section, 50
 - beer mug, 46
 - can, 48
 - cantilever beam, 53–54

- Optimum design problem formulation
 - (*continued*)
 - circular tank, 48
 - cost transportation system, 49
 - crude oil refinery, 46
 - diet of bread and milk, 47
 - electric generators, 49
 - exercises, 46–54
 - hollow circular beam, 54
 - manufacturer's products, 47
 - multistory office building, 46
 - parallel flow heat exchanger, 46–47
 - parking lot, 47
 - shipping container, 48
 - steel framework, 48
 - still in bathtub, 47
 - tonnage of iron ore, 52–53
 - top rectangular container to transport materials, 48
 - two-bar truss, 49–50
 - vegetable oil stocks, 50–51
 - water canal, 53
 - Optimum design problems, classification of
 - mixed variable, 514–515
 - Optimum design problems, constrained, 119–143, 418–420
 - example—constrained minimization problem using `fmincon`, 418–420
 - example—constrained optimum point, 120
 - example—cylindrical tank design, 127
 - example—equality constrained problem, 127
 - example—`fmincon` in Optimization Toolbox, 418–420
 - example—inequality constrained problem, 128–130
 - example—infeasible problem, 121
 - example—Lagrange multipliers and their geometrical meaning, 122–125
 - example—solution of KKT necessary conditions, 134–137, 137–140
 - example—solutions of KKT necessary conditions, 132–134
 - example—unconstrained optimum point for constrained problem, 120
 - example—use of Lagrange multipliers, 127
 - example—use of necessary conditions, 128–130
 - inequality constraints, 128–140
 - KKT, 128–140
 - necessary conditions, 128–140
 - necessary conditions: equality constraints, 121–128
 - role of constraints, 119–121
 - solution of KKT necessary conditions using Excel, 140–141
 - solution of KKT necessary conditions using MATLAB, 141–143
 - Optimum design problems, unconstrained, 103–119
 - Optimum design versus optimal control, 6–7
 - Optimum design with MATLAB, 413–432
 - constrained optimum design problems, 418–420
 - exercises, 429–432
 - Optimization Toolbox, 413–415
 - optimum design examples with MATLAB, 420–429
 - unconstrained optimum design problems, 415–418
 - Optimum designs, multiple, 66
 - Optimum solution, identification of, 63
 - Optimum solutions for LP problems, 201
 - Out-of-plane loads, 481–483
- P
- Parameters, ranging right side, 235–238
 - Pareto fitness function, 554
 - Pareto optimal set, generation of, 551–552
 - Pareto optimality, 548–549
 - weak, 549
 - Pareto-set filter, 554
 - Performance
 - numerical, 585–588
 - of stochastic zooming method, 586–587
 - Performance methods using unconstrained problems, 586
 - Performance requirements, multiple, 483–491
 - Phase I algorithm, 220
 - Phase I problem, definition of, 220
 - Phase II algorithm, 221
 - Physical meaning of Lagrange multipliers, 143–148
 - Physical programming, 551
 - Pivot step, 205–206
 - Plotting
 - of function contours, 64
 - functions, 61–62
 - of objective function contours, 63
 - Points
 - constraint status at design, 342–343
 - sets and, 7–9
 - utopia, 550
 - Points, irregular, 178–179
 - example—check for KKT conditions at irregular points, 178–179
 - Polynomial interpolation, 306–309
 - alternate quadratic interpolation, 308–309
 - quadratic curve fitting, 306–308
 - Population defined, 532

- Postoptimality analysis, 143–148, 228–243
 - changes in coefficient matrix, 241–243
 - changes in resource limits, 229–230
 - constraint variation sensitivity result, 148
 - effect of scaling constraint on Lagrange multiplier, 147–148
 - example— \leq type constraints, 236–237
 - example— \leq type constraints, 239–240
 - example—effect of scaling constraint, 147–148
 - example—effect of scaling cost function, 146–147
 - example—equality and \geq type constraints, 237–238, 240–241
 - example—Lagrange multipliers, 146–147, 147–148
 - example—optimum cost function, 145
 - example—ranges for cost coefficients, 239–240, 240–241
 - example—ranges for resource limits, 236–237, 237–238
 - example—recovery of Lagrange multipliers for \leq type constraint, 230–232
 - example—variations of constraint limits, 145
 - ranging cost coefficients, 239–241
 - ranging right side parameters, 235–238
 - recovery of Lagrange multipliers for \geq type constraints, 232–234
 - scaling cost function on Lagrange multipliers, 146–147
- Potential constraint strategy, 478
- Practical design, issues in, 478–479
- Practical design problems, example of, 467–472
- Practical design problems, formulation of, 466–472
 - example—design of two-member frame, 469–472
 - example of practical design optimization problem, 467–472
 - general guidelines, 466–467
- Preferences and utility functions, 551
- Preparation, interactive data, 448
- Present worth (PW), 598
 - comparisons, 601
- Problem formulation, optimum design, 15–54
 - design of cabinet, 30–32
 - design of can, 18–20
 - design of coil springs, 36–38
 - design of two-bar bracket, 24–29
 - general mathematical model for optimum design, 41–45
 - insulated spherical tank design, 20–22
 - minimum cost cylindrical tank design, 35
 - minimum weight design of symmetric three-bar truss, 38–41
 - minimum weight tubular column design, 32–35
 - problem formulation process, 16–18
 - saw mill operation, 22–23
- Problem formulation, optimum design (exercises), 46–54
 - beam of rectangular cross-section, 50
 - beer mug, 46
 - can, 48
 - cantilever beam, 53–54
 - circular tank, 48
 - cost transportation system, 49
 - crude oil refinery, 46
 - diet of bread and milk, 47
 - electric generators, 49
 - exercises, 46–54
 - hollow circular beam, 54
 - manufacturer's products, 47
 - multistory office building, 46
 - parallel flow heat exchanger, 46–47
 - parking lot, 47
 - shipping container, 48
 - steel framework, 48
 - still in bathtub, 47
 - tonnage of iron ore, 52–53
 - top rectangular container to transport materials, 48
 - two-bar truss, 49–50
 - vegetable oil stocks, 50–51
 - water canal, 53
- Problem formulation process, 16–18
- Problems
 - classification of mixed variable optimum design, 514–515
 - concepts related to algorithms for constrained, 340–342
 - definition of Phase I, 220
 - example of practical design, 467–472
 - formulation of spring design, 454–455
 - graphical solutions for beam design, 69–72
 - infeasible, 67–69
 - integer programming, 32
 - interactive solution for spring design, 455–457
 - linear programming, 56, 191
 - minimum control effort, 503–505
 - minimum time control, 505–508
 - MV-OPT, 514
 - numerical methods for nonlinear optimization, 277
 - optimum solution for spring design, 455
 - optimum solutions for LP, 201
 - profit maximization, 55–56
 - prototype optimal control, 493–497
 - solution of constrained, 332–334

- Problems (*continued*)
 - solution of LP, 243–246
 - sufficiency check for rectangular beam design, 184–185
 - with unbounded solutions, 66–67
 - See also* Subproblems
- Problems, concepts related to linear programming, 195–201
 - basic concepts, 195–196
 - example—characterization of solution for LP problems, 196–198
 - example—determination of basic solutions, 199–200
 - example—profit maximization problem, 196–198
 - LP terminology, 198–200
 - optimum solutions for LP problems, 201
- Problems, constrained optimum design, 119–143, 418–420
 - example—constrained minimization problem using *fmincon*, 418–420
 - example—constrained optimum point, 120
 - example—cylindrical tank design, 127
 - example—equality constrained problem, 127
 - example—*fmincon* in Optimization Toolbox, 418–420
 - example—inequality constrained problem, 128–130
 - example—infeasible problem, 121
 - example—Lagrange multipliers and their geometrical meaning, 122–125
 - example—solution of KKT necessary conditions, 134–137, 137–140
 - example—solutions of KKT necessary conditions, 132–134
 - example—unconstrained optimum point for constrained problem, 120
 - example—use of Lagrange multipliers, 127
 - example—use of necessary conditions, 128–130
 - inequality constraints, 128–140
 - KKT, 128–140
 - necessary conditions, 128–140
 - necessary conditions: equality constraints, 121–128
 - role of constraints, 119–121
 - solution of KKT necessary conditions using Excel, 140–141
 - solution of KKT necessary conditions using MATLAB, 141–143
- Problems, convex programming, 153–156, 157–158
- Problems, definition of standard linear programming, 192–195
 - example—conversion to standard LP form, 194–195
 - linear constraints, 192–193
 - standard LP definition, 193–194
 - unrestricted variables, 193
- Problems, formulation of practical design optimization, 466–472
 - example—design of two-member frame, 469–472
 - example of practical design optimization problem, 467–472
 - general guidelines, 466–467
- Problems, GA for sequencing-type, 538–539
 - example—bolt insertion sequence determination, 539
- Problems, global optimization of structural design, 587–588
- Problems, linearization of constrained, 346–352
 - example—definition of linearized subproblem, 348–350
 - example—linearization of rectangular beam design problem, 350–352
- Problems, performance methods using unconstrained, 586
- Problems, QP, 383–388
 - definition of QP problem, 383–384
 - example—solution of QP problem, 386–388
 - KKT necessary conditions for QP problem, 384
 - Simplex method for solving QP problem, 385–386
 - transformation of KKT conditions, 384–385
- Problems, unconstrained design, 103–119, 415–418
 - concepts relating to optimality conditions, 103–104
 - example—adding constant to function, 111
 - example—cylindrical tank design, 113–114
 - example—determination of local minimum points, 105–106
 - example—effects of scaling, 111
 - example—local minima for function of two variables, 112–113, 115–116
 - example—local minimum points using necessary conditions, 107–108
 - example—minimum cost spherical tank using necessary conditions, 108–109
 - example—multivariable unconstrained minimization, 416–418
 - example—numerical solution of necessary conditions, 114–115
 - example—single-variable unconstrained minimization, 415–416
 - example—using necessary conditions, 105–106, 113–114

- example—using optimality conditions, 112–113, 115–116
 - optimality conditions for functions of several variables, 109–116
 - optimality conditions for functions of single variables, 104–109
 - roots of nonlinear equations using Excel, 116–119
 - Procedures, Gaussian elimination, 621–625
 - Procedures, gradient evaluation requires special, 465
 - Process, design, 2–4
 - Process, problem formulation, 16–18
 - data and information collection, 16
 - identification/definition of design variables, 16–17
 - identification of constraints, 17–18
 - identification of criterion to be optimized, 17
 - project/problem statement, 16
 - Profit maximization problem, 55–56
 - Programming
 - duality in linear, 263–274
 - goal, 559
 - physical, 551
 - Programming, control of systems by nonlinear, 493–508
 - comparison of three formulations, 508
 - minimization of errors in state variables, 497–503
 - minimum control effort problem, 503–505
 - minimum time control problem, 505–508
 - optimal control of system motion, 508
 - prototype optimal control problem, 493–497
 - Programming problems
 - convex, 153–156, 157–158
 - linear, 56, 191, 195–201
 - Programs, sample computer, 657–673
 - equal interval search, 657–660
 - golden section search, 660
 - modified Newton's method, 669–673
 - steepest descent search, 660–669
 - Projection method, gradient, 409–410
 - Prototype optimal control problem, 493–497
 - Pure random search, 573
 - PW. *See* Present worth
- Q**
- QP. *See* Quadratic programming
 - Quadratic curve fitting, 306–308
 - Quadratic forms and definite matrices, 96–102
 - example—calculations for gradient of quadratic form, 102
 - example—calculations for Hessian of quadratic form, 102
 - example—determination of form of matrix, 99, 100–101
 - example—matrix of quadratic form, 98
 - Quadratic interpolation, alternate, 308–309
 - Quadratic programming (QP) problem, 383–388
 - definition of QP problem, 383–384
 - example—solution of QP problem, 386–388
 - KKT necessary conditions for QP problem, 384
 - Simplex method for solving QP problem, 385–386
 - transformation of KKT conditions, 384–385
 - Quadratic programming (QP) subproblem, 358–363
 - definition of QP subproblem, 358–361
 - deviation of, 400–402
 - example—definition of QP subproblem, 359–361
 - example—solution of QP subproblem, 362–363
 - solution of QP subproblem, 361–363
 - Quasi-Newton Hessian approximation, 403–404
 - Quasi-Newton methods, 324–328
 - BFGS method, 327–328
 - DFP method, 324–327
 - direct Hessian updating, 327–328
 - example—application of BFGS method, 328
 - example—application of DFP method, 325–327
 - inverse Hessian updating, 324–327
 - observations on constrained, 406
 - Quasi-Newton methods, constrained, 400–407
 - descent functions, 406–407
 - deviation of QP subproblem, 400–402
 - example—use of constrained quasi-Newton method, 404–405
 - modified constrained steepest descent algorithm, 404–405
 - observations on constrained quasi-Newton methods, 406
- R**
- Random search, pure, 573
 - Ranging cost coefficients, 239–241
 - Ranging right side parameters, 235–238
 - Rectangular beam, design of, 162–166
 - Rectangular beam design problem, sufficiency check for, 184–185
 - Recursive quadratic programming (RQP), 400
 - Reducing methods, interval, 286
 - Reduction algorithm, cost, 436–440
 - Regions
 - identification and hatching of infeasible, 62
 - identification of feasible, 62–63

- Representation, design, 532
- Reproduction defined, 534
- Requirements, multiple performance, 483–491
- Right side parameters, ranging, 235–238
- Robust algorithms, 478
- Rounding-off method, dynamic, 524–525
- Row matrix, 613
- Row vector, 613

- S
- SA. *See* Simulated annealing
- Saw mill operation, 22–23
- Scalar matrix, 616
- Scalar operation, 414
- Scalarization methods, 551
- Scaling of design variables, 315–318
 - example—effect of scaling of design variables, 315–317, 317–318
- Search direction determination, 318–324, 324–328
- Search method, neighborhood, 525
- Searches
 - alternate equal interval, 288–289
 - equal interval, 286–288, 657–660
 - golden section, 289–293, 660
 - inaccurate line, 309–310
 - pure random, 573
 - steepest descent, 660–669
- Second-order information, 179
- Second-ordered conditions for constrained optimization, 179–184
- Section search, golden, 660
- Sequencing-type problems, GA for, 538–539
- Sequential linear programming (SLP) algorithm, 352–358
 - basic idea—move limits, 352–353
 - example—sequential linear programming algorithm, 354–355
 - example—use of sequential linear programming, 356–357
 - SLP algorithm, 353–357
 - SLP algorithm observations, 357–358
- Sequential linearization methods, 522
- Sequential quadratic programming (SQP), 364, 400, 587
 - option, 482
- Sequential unconstrained minimization techniques, 333–334
- Set, generation of Pareto optimal, 551–552
- Sets and points, 7–9
- Sets, convex, 149–151
- SI units, U.S.-British versus, 12–14
- Simplex algorithms, 211–212
- Simplex in two-dimensional space, 202
- Simplex method
 - basic steps of, 206–211
 - two-phase, 218–228
- Simplex method, alternate, 262–263
 - example—Big-M method for equality and \geq type constraints, 262–263
- Simplex method, basic ideas and steps of, 201–218
 - basic steps of Simplex method, 206–211
 - canonical form/general solution of $Ax = b$, 202–203
 - example—canonical form and tableau, 204
 - example—identification of unbounded problem with Simplex method, 218
 - example—LP problem with multiple solutions, 216–217
 - example—pivot step, 205–206
 - example—solution by Simplex method, 213–214
 - example—solution of profit maximization problem, 214–216
 - interchange of basic and nonbasic variables, 205–206
 - pivot step, 205–206
- Simplex, 202
- Simplex algorithms, 211–212
- steps of Simplex method, 207–211
- tableau, 203–204
- Simplex method, derivation of, 259–261
 - selection of basic variable, 259–261
 - selection of nonbasic variable, 260–261
- Simplex method for solving QP problem, 385–386
- Simulated annealing (SA), 522–524, 586, 587
- Single nonlinear equation, 647–650
- Single variables, optimality conditions for functions of, 104–109
- SLP. *See* Sequential linear programming
- Software, general-purpose, 479–481
 - integration of application into general purpose software, 480–481
 - software selection, 480
- Software, integration of application into general purpose, 480–481
- Software, interactive design optimization, 450–454
 - capabilities of IDESIGN, 453–454
 - user interface for IDESIGN, 451–452
- Software selection, 480
- Solution, compromise, 550
- Solution concepts, 515–516, 548–550
 - compromise solution, 550
 - efficiency and dominance, 549–550
 - Pareto optimality, 548–549

- utopia point, 550
 - weak Pareto optimality, 549
- Solution methods, basic concepts of, 565–567
- Solutions
 - degenerate basic feasible, 226–228
 - design problem with multiple, 66
 - identification of optimum, 63
 - problem with unbounded, 66–67
- Spaces
 - criterion, 546–548
 - design, 546–548
 - Simplex in two-dimensional, 202
 - vector, 639–641
- Special procedures, gradient evaluation requires, 465
- Spherical tank design, insulated, 20–22
- Spring design problem
 - formulation of, 454–455
 - interactive solution for, 455–457
 - optimum solution for, 455
- Springs, design of coil, 36–38
- SQP. *See* Sequential quadratic programming
- Square matrices, 616–617
- Standard linear programming problem, definition of, 192–195
- Standard LP definition, 193–194
- State variables, minimization of errors in, 497–503
 - discussion of results, 502–503
 - effect of problem normalization, 500–502
 - formulation for numerical solution, 498–499
 - numerical results, 499–500
- Steepest descent algorithm, modified constrained, 404–405
- Steepest descent directions, orthogonality of, 314–315
- Steepest descent method, 293–296, 310–315
 - example—use of steepest descent algorithm, 294–295, 295–296
 - example—verification of properties of gradient vector, 312–314
 - orthogonality of steepest descent directions, 314–315
 - properties of gradient vector, 310–314
- Steepest descent method, constrained, 363–369
 - CSD algorithm observations, 368–369
 - CSD algorithm, 368
 - descent function, 364–365
 - example—calculation of descent function, 365, 366–367
 - example—golden section search, 366–367
 - step size determination, 366–367
- Steepest descent search, 660–669
- Step size determination, 305–310
 - example—alternate quadratic interpolation, 308–309
 - example—one-dimensional minimization, 307–308, 308–309
 - inaccurate line search, 309–310
 - polynomial interpolation, 306–309
- Step size determination, approximate, 388–399
 - basic idea, 388–389
 - CSD algorithm with appropriate step size, 393–399
 - descent condition, 389–393
 - example—calculations for step size, 391–393
 - example—constrained steepest descent method, 391–393
 - example—effect of γ on performance of CSD algorithm, 396–397
 - example—minimum area design of rectangular beam, 398–399
 - example—penalty parameter R and CSD algorithm, 397–398
 - example—use of constrained steepest descent algorithm, 393–396
- Step size determination, ideas and algorithms for, 282–293
 - alternate equal interval search, 288–289
 - analytical method to compare step size, 283–285
 - definition of one-dimensional minimization subproblem, 282–283
 - equal interval search, 286–288
 - example—analytical step size determination, 284–285
 - example—minimization of function by golden section search, 292–293
 - golden section search, 289–293
 - numerical methods and compute step size, 285–286
- Steps
 - descent, 280–281
 - pivot, 205–206
- Stochastic integration, 579
- Stochastic methods, 572–579
 - A-R methods, 578
 - clustering methods, 573–575
 - CRS method, 575–578
 - multistart method, 573
 - pure random search, 573
 - stochastic integration, 579
- Stochastic methods, two local-global, 579–585
 - conceptual local-global algorithm, 579–580
 - domain elimination method, 580–582
 - operations analysis of methods, 583–585
 - stochastic zooming method, 582–583
- Stochastic zooming method, 582–583

- Stochastic zooming method, performance of, 586–587
- Strategy, potential constraint, 379–382, 478
 example—determination of potential constraint set, 380–381
 example—search direction and potential constraint strategy, 381–382
- Structural design problems, optimization of, 587–588
- Structures
 asymmetric three-bar, 484–490
 symmetric three-bar, 483–484
- Structures, optimum design of three-bar, 483–491
 asymmetric three-bar structure, 484–490
 comparison of solutions, 490–491
 symmetric three-bar structure, 483–484
- Subproblem, deviation of QP, 400–402
- Subproblems, QP, 358–363
 definition of QP subproblem, 358–361
 example—solution of QP subproblem, 362–363
 examples—definition of QP subproblem, 359–361
 solution of QP subproblem, 361–363
- Subscripts. *See* Superscripts/subscripts
- Summation notation, superscripts/subscripts and, 9–10
- Superscripts/subscripts and summation notation, 9–10
- Symmetric three-bar structure, 483–484
- Symmetric three-bar truss, minimum weight design of, 38–41
- System motion, optimal control of, 508
- Systems, linear, 618–619
- Systems, optimal control, 493–508
- T**
- Tableau, defined, 203
- Tangent hyperplane, constraint, 179
- Tank design
 cylindrical, 35
 insulated spherical, 20–22
- Taylor's expansion, 93–96
- Techniques
 niche, 555
 sequential unconstrained minimization, 333–334
- Terminology and notation
 basic, 7–14
- Terminology and notations, basic
 functions, 11–12
 norm/length of vectors, 11
 notation for constraints, 9
 sets and points, 7–9
 superscripts/subscripts and summation notation, 9–10
 U.S.-British versus SI units, 12–14
- Terminology, LP, 198–200
- Three-bar structure, asymmetric, 484–490
- Three-bar structure, optimum design of, 483–491
 asymmetric three-bar structure, 484–490
 comparison of solutions, 490–491
 symmetric three-bar structure, 483–484
- Three-bar truss, minimum weight design of
 symmetric, 38–41
- Time control problem, minimum, 505–508
- Time value of money, 593–598
- Toolbox, Optimization, 413–415
 array operation, 414
 matrix operation, 414
 scalar operation, 414
 variables and expressions, 413–414
- Triangle matrix
 lower, 616
 upper, 616
- Truss, minimum weight design of symmetric three-bar, 38–41
- Tubular column design, minimum weight, 32–35
- Tubular column, graphical solution for minimum weight, 69
- Tunneling method, 571–572
- Two-bar bracket, design of, 24–29
 example—optimum design of two-bar bracket, 28–29
- Two-dimensional space, Simplex in, 202
- Two-member frame, optimum design of, 481–483
- Two-phase Simplex method, 218–228
- U**
- Unbounded solution, problem with, 66–67
- Unconstrained methods, engineering applications of, 329–332
 example—minimization of total potential energy of two-bar truss, 330–331
 example—roots of nonlinear equations, 331–332
 example—unconstrained minimization, 331–332
 minimization of total potential energy, 329–331
 solution of nonlinear equations, 331–332
- Unconstrained minimization techniques, sequential, 333–334
- Unconstrained optimization methods, 332–334
 augmented Lagrangian methods, 334

- multiplier methods, 334
 - sequential unconstrained minimization techniques, 333–334
 - Unconstrained optimum design, numerical methods for, 277–304, 305–337
 - concepts related to numerical algorithms, 278–282
 - conjugate gradient method, 296–299
 - ideas and algorithms for step size determination, 282–293
 - Newton's method, 318–324
 - quasi-Newton methods, 324–328
 - scaling of design variables, 315–318
 - search direction determination, 293–296, 296–299, 318–324, 324–328
 - solution of constrained problems, 332–334
 - steepest descent method, 293–296, 310–315
 - step size determination, 305–310
 - unconstrained optimization methods, 332–334
 - Unconstrained optimum design, numerical methods for, engineering applications of unconstrained methods, 329–332
 - Unconstrained optimum design, numerical methods for (exercises), 300–304
 - Unconstrained optimum design, numerical methods for, exercises, 335–337
 - Unconstrained optimum design, numerical methods for (exercises)
 - basic ideas and algorithm for step size determination, 300–302
 - conjugate gradient method, 303
 - exercises, 300–304
 - general concepts related to numerical algorithm, 300
 - search direction determination, 302–303
 - steepest descent method, 302–303
 - Unconstrained optimum design problems, 103–119, 415–418
 - example—multivariable unconstrained minimization, 416–418
 - example—single-variable unconstrained minimization, 415–416
 - Unconstrained problems, performance methods using, 586
 - Unknowns, solution of m linear equations in n , 628–635
 - Unrestricted variables, 193
 - Upper triangle matrix, 616
 - U.S.-British versus SI units, 12–14
 - User interface for IDESIGN, 451–452
 - Utility functions, preferences and, 551
 - Utopia point, 550
- V
- Variable optimization, continuous, 492
 - Variable optimization, discrete, 492–493
 - Variable optimum design, discrete, 491–493
 - Variables
 - binary, 513
 - discrete, 513
 - and expressions, 413–414
 - integer, 513
 - linked discrete, 513
 - methods for linked discrete, 525–526
 - unrestricted, 193
 - Variables, artificial, 218–228
 - artificial cost function, 219–220
 - definition of Phase I problem, 220
 - degenerate basic feasible solution, 226–228
 - example—feasible problem, 223–224
 - example—implications of degenerate feasible solution, 226–228
 - example—unbounded problem, 225–226
 - example—use of artificial variables, 225–226
 - example—use of artificial variables for \geq type constraints, 221–223
 - example—use of artificial variables for equality constraints, 223–224
 - Phase I algorithm, 220
 - Phase II algorithm, 221
 - Variables, minimization of errors in state, 497–503
 - discussion of results, 502–503
 - effect of problem normalization, 500–502
 - formulation for numerical solution, 498–499
 - numerical results, 499–500
 - Variables, optimality conditions for functions of single, 104–109
 - Variables, scaling of design, 315–318
 - example—effect of scaling of design variables, 315–317, 317–318
 - Vector and matrix algebra, 611–646
 - concepts related to set of vectors, 635–641
 - definition of matrices, 611–613
 - eigenvalues and eigenvectors, 642–643
 - exercises, 645–646
 - norm and condition number of matrix, 643–644
 - solution of m linear equations in n unknowns, 628–635
 - solution of n linear equations in n unknowns, 618–628
 - type of matrices and their operations, 613–618
 - Vector evaluated genetic algorithm (VEGA), 553
 - Vector, gradient, 90–91

- Vector methods, 551
- Vector spaces, 639–641
 - vector spaces, 639–641
- VEGA. *See* Vector evaluated genetic algorithm
- Vectors, 613
 - column, 613
 - norm/length of, 11
 - properties of gradient, 310–314
 - row, 613
- Vectors, concepts related to set of, 635–641
 - example—checking for linear independence of vectors, 636–637
 - example—checking for vector spaces, 639–641
 - linear independence of set of vectors, 635–639
- W
 - Wall bracket, design of, 158–162
 - Weighted global criterion method, 556–558
 - Weighted min-max method, 556
 - Weighted sum method, 555–556
- Z
 - Zooming methods, 568–569
 - performances of stochastic, 586–587
 - stochastic, 582–583