

Wiley Series in Discrete Mathematics and Optimization

AN
INTRODUCTION
TO
OPTIMIZATION

FOURTH EDITION

Edwin K. P. Chong
Stanislaw H. Żak

 **WILEY**

WWW.
LINK AVAILABLE

AN INTRODUCTION TO OPTIMIZATION

**WILEY SERIES IN
DISCRETE MATHEMATICS AND OPTIMIZATION**

A complete list of titles in this series appears at the end of this volume.

AN INTRODUCTION TO OPTIMIZATION

Fourth Edition

Edwin K. P. Chong

Colorado State University

Stanislaw H. Zak

Purdue University



WILEY

A JOHN WILEY & SONS, INC., PUBLICATION

Copyright © 2013 by John Wiley & Sons, Inc. All rights reserved

Published by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 750-4470, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permission>.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services or for technical support, please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic formats. For more information about Wiley products, visit our web site at www.wiley.com.

Library of Congress Cataloging-in-Publication Data

Chong, Edwin Kah Pin.

An introduction to optimization / Edwin K. P. Chong, Colorado State University, Stanislaw H. Zak, Purdue University. — Fourth edition.

pages cm

Summary: "The purpose of the book is to give the reader a working knowledge of optimization theory and methods" — Provided by publisher.

Includes bibliographical references and index.

ISBN 978-1-118-27901-4 (hardback)

1. Mathematical optimization. I. Zak, Stanislaw H. II. Title.

QA402.5.C476 2012

519.6—dc23

2012031772

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

*To my wife, Yat-Yee,
and to my parents, Paul
and Julianne Chong.
Edwin K. P. Chong*

*To JMJ; my wife,
Mary Ann; and my
parents, Janina and
Konstanty Żak.
Stanislaw H. Żak*

CONTENTS

Preface	xiii
---------	------

PART I MATHEMATICAL REVIEW

1 Methods of Proof and Some Notation	3
1.1 Methods of Proof	3
1.2 Notation	5
Exercises	6
2 Vector Spaces and Matrices	7
2.1 Vector and Matrix	7
2.2 Rank of a Matrix	13
2.3 Linear Equations	17
2.4 Inner Products and Norms	19
Exercises	22
3 Transformations	25
3.1 Linear Transformations	25

3.2	Eigenvalues and Eigenvectors	26
3.3	Orthogonal Projections	29
3.4	Quadratic Forms	31
3.5	Matrix Norms	35
	Exercises	40
4	Concepts from Geometry	45
4.1	Line Segments	45
4.2	Hyperplanes and Linear Varieties	46
4.3	Convex Sets	48
4.4	Neighborhoods	50
4.5	Polytopes and Polyhedra	52
	Exercises	53
5	Elements of Calculus	55
5.1	Sequences and Limits	55
5.2	Differentiability	62
5.3	The Derivative Matrix	63
5.4	Differentiation Rules	67
5.5	Level Sets and Gradients	68
5.6	Taylor Series	72
	Exercises	77
PART II UNCONSTRAINED OPTIMIZATION		
6	Basics of Set-Constrained and Unconstrained Optimization	81
6.1	Introduction	81
6.2	Conditions for Local Minimizers	83
	Exercises	93
7	One-Dimensional Search Methods	103
7.1	Introduction	103
7.2	Golden Section Search	104
7.3	Fibonacci Method	108
7.4	Bisection Method	116
7.5	Newton's Method	116
7.6	Secant Method	120
7.7	Bracketing	123

7.8	Line Search in Multidimensional Optimization	124
	Exercises	126
8	Gradient Methods	131
8.1	Introduction	131
8.2	The Method of Steepest Descent	133
8.3	Analysis of Gradient Methods	141
	Exercises	153
9	Newton's Method	161
9.1	Introduction	161
9.2	Analysis of Newton's Method	164
9.3	Levenberg-Marquardt Modification	168
9.4	Newton's Method for Nonlinear Least Squares	168
	Exercises	171
10	Conjugate Direction Methods	175
10.1	Introduction	175
10.2	The Conjugate Direction Algorithm	177
10.3	The Conjugate Gradient Algorithm	182
10.4	The Conjugate Gradient Algorithm for Nonquadratic Problems	186
	Exercises	189
11	Quasi-Newton Methods	193
11.1	Introduction	193
11.2	Approximating the Inverse Hessian	194
11.3	The Rank One Correction Formula	197
11.4	The DFP Algorithm	202
11.5	The BFGS Algorithm	207
	Exercises	211
12	Solving Linear Equations	217
12.1	Least-Squares Analysis	217
12.2	The Recursive Least-Squares Algorithm	227
12.3	Solution to a Linear Equation with Minimum Norm	231
12.4	Kaczmarz's Algorithm	232
12.5	Solving Linear Equations in General	236

Exercises	244
13 Unconstrained Optimization and Neural Networks	253
13.1 Introduction	253
13.2 Single-Neuron Training	256
13.3 The Backpropagation Algorithm	258
Exercises	270
14 Global Search Algorithms	273
14.1 Introduction	273
14.2 The Nelder-Mead Simplex Algorithm	274
14.3 Simulated Annealing	278
14.4 Particle Swarm Optimization	282
14.5 Genetic Algorithms	285
Exercises	298
PART III LINEAR PROGRAMMING	
15 Introduction to Linear Programming	305
15.1 Brief History of Linear Programming	305
15.2 Simple Examples of Linear Programs	307
15.3 Two-Dimensional Linear Programs	314
15.4 Convex Polyhedra and Linear Programming	316
15.5 Standard Form Linear Programs	318
15.6 Basic Solutions	324
15.7 Properties of Basic Solutions	327
15.8 Geometric View of Linear Programs	330
Exercises	335
16 Simplex Method	339
16.1 Solving Linear Equations Using Row Operations	339
16.2 The Canonical Augmented Matrix	346
16.3 Updating the Augmented Matrix	349
16.4 The Simplex Algorithm	350
16.5 Matrix Form of the Simplex Method	357
16.6 Two-Phase Simplex Method	361
16.7 Revised Simplex Method	364
Exercises	369

17 Duality	379
17.1 Dual Linear Programs	379
17.2 Properties of Dual Problems	387
Exercises	394
18 Nonsimplex Methods	403
18.1 Introduction	403
18.2 Khachiyan's Method	405
18.3 Affine Scaling Method	408
18.4 Karmarkar's Method	413
Exercises	426
19 Integer Linear Programming	429
19.1 Introduction	429
19.2 Unimodular Matrices	430
19.3 The Gomory Cutting-Plane Method	437
Exercises	447
PART IV NONLINEAR CONSTRAINED OPTIMIZATION	
20 Problems with Equality Constraints	453
20.1 Introduction	453
20.2 Problem Formulation	455
20.3 Tangent and Normal Spaces	456
20.4 Lagrange Condition	463
20.5 Second-Order Conditions	472
20.6 Minimizing Quadratics Subject to Linear Constraints	476
Exercises	481
21 Problems with Inequality Constraints	487
21.1 Karush-Kuhn-Tucker Condition	487
21.2 Second-Order Conditions	496
Exercises	501
22 Convex Optimization Problems	509
22.1 Introduction	509
22.2 Convex Functions	512
22.3 Convex Optimization Problems	521

22.4	Semidefinite Programming	527
	Exercises	540
23	Algorithms for Constrained Optimization	549
23.1	Introduction	549
23.2	Projections	549
23.3	Projected Gradient Methods with Linear Constraints	553
23.4	Lagrangian Algorithms	557
23.5	Penalty Methods	564
	Exercises	571
24	Multiobjective Optimization	577
24.1	Introduction	577
24.2	Pareto Solutions	578
24.3	Computing the Pareto Front	581
24.4	From Multiobjective to Single-Objective Optimization	585
24.5	Uncertain Linear Programming Problems	588
	Exercises	596
References		599
Index		609

PREFACE

Optimization is central to any problem involving decision making, whether in engineering or in economics. The task of decision making entails choosing among various alternatives. This choice is governed by our desire to make the “best” decision. The measure of goodness of the alternatives is described by an objective function or performance index. Optimization theory and methods deal with selecting the best alternative in the sense of the given objective function.

The area of optimization has received enormous attention in recent years, primarily because of the rapid progress in computer technology, including the development and availability of user-friendly software, high-speed and parallel processors, and artificial neural networks. A clear example of this phenomenon is the wide accessibility of optimization software tools such as the Optimization Toolbox of MATLAB¹ and the many other commercial software packages.

There are currently several excellent graduate textbooks on optimization theory and methods (e.g., [3], [39], [43], [51], [87], [88], [104], [129]), as well as undergraduate textbooks on the subject with an emphasis on engineering design (e.g., [1] and [109]). However, there is a need for an introductory

¹MATLAB is a registered trademark of The MathWorks, Inc.

textbook on optimization theory and methods at a senior undergraduate or beginning graduate level. The present text was written with this goal in mind. The material is an outgrowth of our lecture notes for a one-semester course in optimization methods for seniors and beginning graduate students at Purdue University, West Lafayette, Indiana. In our presentation, we assume a working knowledge of basic linear algebra and multivariable calculus. For the reader's convenience, a part of this book (Part I) is devoted to a review of the required mathematical background material. Numerous figures throughout the text complement the written presentation of the material. We also include a variety of exercises at the end of each chapter. A solutions manual with complete solutions to the exercises is available from the publisher to instructors who adopt this text. Some of the exercises require using MATLAB. The student edition of MATLAB is sufficient for all of the MATLAB exercises included in the text. The MATLAB source listings for the MATLAB exercises are also included in the solutions manual.

The purpose of the book is to give the reader a working knowledge of optimization theory and methods. To accomplish this goal, we include many examples that illustrate the theory and algorithms discussed in the text. However, it is not our intention to provide a cookbook of the most recent numerical techniques for optimization; rather, our goal is to equip the reader with sufficient background for further study of advanced topics in optimization.

The field of optimization is still a very active research area. In recent years, various new approaches to optimization have been proposed. In this text, we have tried to reflect at least some of the flavor of recent activity in the area. For example, we include a discussion of randomized search methods—these include particle swarm optimization and genetic algorithms, topics of increasing importance in the study of complex adaptive systems. There has also been a recent surge of applications of optimization methods to a variety of new problems. An example of this is the use of descent algorithms for the training of feedforward neural networks. An entire chapter in the book is devoted to this topic. The area of neural networks is an active area of ongoing research, and many books have been devoted to this subject. The topic of neural network training fits perfectly into the framework of unconstrained optimization methods. Therefore, the chapter on feedforward neural networks not only provides an example of application of unconstrained optimization methods but also gives the reader an accessible introduction to what is currently a topic of wide interest.

The material in this book is organized into four parts. Part I contains a review of some basic definitions, notations, and relations from linear algebra, geometry, and calculus that we use frequently throughout the book. In Part II we consider unconstrained optimization problems. We first discuss some theoretical foundations of set-constrained and unconstrained optimization, including necessary and sufficient conditions for minimizers and maximizers. This is followed by a treatment of various iterative optimization algorithms, including line search methods, together with their properties. A discussion of global

search algorithms is included in this part. We also analyze the least-squares optimization problem and the associated recursive least-squares algorithm. Parts III and IV are devoted to constrained optimization. Part III deals with linear programming problems, which form an important class of constrained optimization problems. We give examples and analyze properties of linear programs, and then discuss the simplex method for solving linear programs. We also provide a brief treatment of dual linear programming problems. We then describe some nonsimplex algorithms for solving linear programs: Khachiyan's method, the affine scaling method, and Karmarkar's method. We wrap up Part III by discussing integer linear programming problems. In Part IV we treat nonlinear constrained optimization. Here, as in Part II, we first present some theoretical foundations of nonlinear constrained optimization problems, including convex optimization problems. We then discuss different algorithms for solving constrained optimization problems. We also treat multiobjective optimization.

Although we have made every effort to ensure an error-free text, we suspect that some errors remain undetected. For this purpose, we provide online updated errata that can be found at the Web site for the book, accessible via

<http://www.wiley.com/mathematics>

We are grateful to several people for their help during the course of writing this book. In particular, we thank Dennis Goodman of Lawrence Livermore Laboratories for his comments on early versions of Part II and for making available to us his lecture notes on nonlinear optimization. We thank Moshe Kam of Drexel University for pointing out some useful references on nonsimplex methods. We are grateful to Ed Silverman and Russell Quong for their valuable remarks on Part I of the first edition. We also thank the students of ECE 580 at Purdue University and ECE 520 and MATH 520 at Colorado State University for their many helpful comments and suggestions. In particular, we are grateful to Christopher Taylor for his diligent proofreading of early manuscripts of this book. This fourth edition incorporates many valuable suggestions of users of the first, second, and third editions, to whom we are grateful.

E. K. P. CHONG AND S. H. ŽAK
Fort Collins, Colorado, and West Lafayette, Indiana

PART I

MATHEMATICAL REVIEW

CHAPTER 1

METHODS OF PROOF AND SOME NOTATION

1.1 Methods of Proof

Consider two statements, “A” and “B,” which could be either true or false. For example, let “A” be the statement “John is an engineering student,” and let “B” be the statement “John is taking a course on optimization.” We can combine these statements to form other statements, such as “A and B” or “A or B.” In our example, “A and B” means “John is an engineering student, and he is taking a course on optimization.” We can also form statements such as “not A,” “not B,” “not (A and B),” and so on. For example, “not A” means “John is not an engineering student.” The truth or falsity of the combined statements depend on the truth or falsity of the original statements, “A” and “B.” This relationship is expressed by means of truth tables; see Tables 1.1 and 1.2.

From the tables, it is easy to see that the statement “not (A and B)” is equivalent to “(not A) or (not B)” (see Exercise 1.3). This is called *DeMorgan’s law*.

In proving statements, it is convenient to express a combined statement by a *conditional*, such as “A implies B,” which we denote “ $A \Rightarrow B$.” The conditional

Table 1.1 Truth Table for “A and B” and “A or B”

A	B	A and B	A or B
F	F	F	F
F	T	F	T
T	F	F	T
T	T	T	T

Table 1.2 Truth Table for “not A”

A	not A
F	T
T	F

Table 1.3 Truth Table for Conditionals and Biconditionals

A	B	$A \Rightarrow B$	$A \Leftarrow B$	$A \Leftrightarrow B$
F	F	T	T	T
F	T	T	F	F
T	F	F	T	F
T	T	T	T	T

“ $A \Rightarrow B$ ” is simply the combined statement “(not A) or B” and is often also read “A only if B,” or “if A then B,” or “A is sufficient for B,” or “B is necessary for A.”

We can combine two conditional statements to form a *biconditional* statement of the form “ $A \Leftrightarrow B$,” which simply means “ $(A \Rightarrow B)$ and $(B \Rightarrow A)$.” The statement “ $A \Leftrightarrow B$ ” reads “A if and only if B,” or “A is equivalent to B,” or “A is necessary and sufficient for B.” Truth tables for conditional and biconditional statements are given in Table 1.3.

It is easy to verify, using the truth table, that the statement “ $A \Rightarrow B$ ” is equivalent to the statement “ $(\text{not } B) \Rightarrow (\text{not } A)$.” The latter is called the *contrapositive* of the former. If we take the contrapositive to DeMorgan’s law, we obtain the assertion that “not (A or B)” is equivalent to “(not A) and (not B).”

Most statements we deal with have the form “ $A \Rightarrow B$.” To prove such a statement, we may use one of the following three different techniques:

1. The direct method

2. Proof by contraposition
3. Proof by contradiction or *reductio ad absurdum*

In the case of the *direct method*, we start with “A,” then deduce a chain of various consequences to end with “B.”

A useful method for proving statements is *proof by contraposition*, based on the equivalence of the statements “ $A \Rightarrow B$ ” and “ $(\text{not } B) \Rightarrow (\text{not } A)$.” We start with “not B,” then deduce various consequences to end with “not A” as a conclusion.

Another method of proof that we use is *proof by contradiction*, based on the equivalence of the statements “ $A \Rightarrow B$ ” and “not (A and (not B)).” Here we begin with “A and (not B)” and derive a contradiction.

Occasionally, we use the *principle of induction* to prove statements. This principle may be stated as follows. Assume that a given property of positive integers satisfies the following conditions:

- The number 1 possesses this property.
- If the number n possesses this property, then the number $n + 1$ possesses it too.

The principle of induction states that under these assumptions any positive integer possesses the property.

The principle of induction is easily understood using the following intuitive argument. If the number 1 possesses the given property, then the second condition implies that the number 2 possesses the property. But, then again, the second condition implies that the number 3 possesses this property, and so on. The principle of induction is a formal statement of this intuitive reasoning.

For a detailed treatment of different methods of proof, see [130].

1.2 Notation

Throughout, we use the following notation. If X is a set, then we write $x \in X$ to mean that x is an element of X . When an object x is not an element of a set X , we write $x \notin X$. We also use the “curly bracket notation” for sets, writing down the first few elements of a set followed by three dots. For example, $\{x_1, x_2, x_3, \dots\}$ is the set containing the elements x_1, x_2, x_3 , and so on. Alternatively, we can explicitly display the law of formation. For example, $\{x : x \in \mathbb{R}, x > 5\}$ reads “the set of all x such that x is real and x is greater than 5.” The colon following x reads “such that.” An alternative notation for the same set is $\{x \in \mathbb{R} : x > 5\}$.

If X and Y are sets, then we write $X \subset Y$ to mean that every element of X is also an element of Y . In this case, we say that X is a *subset* of Y . If X and Y are sets, then we denote by $X \setminus Y$ (“ X minus Y ”) the set of all points in X that are not in Y . Note that $X \setminus Y$ is a subset of X . The

notation $f : X \rightarrow Y$ means “ f is a function from the set X into the set Y .” The symbol $:=$ denotes arithmetic assignment. Thus, a statement of the form $x := y$ means “ x becomes y .” The symbol \triangleq means “equals by definition.”

Throughout the text, we mark the end of theorems, lemmas, propositions, and corollaries using the symbol \square . We mark the end of proofs, definitions, and examples by \blacksquare .

We use the IEEE style when citing reference items. For example, [77] represents reference number 77 in the list of references at the end of the book.

EXERCISES

1.1 Construct the truth table for the statement “ $(\text{not } B) \Rightarrow (\text{not } A)$,” and use it to show that this statement is equivalent to the statement “ $A \Rightarrow B$.”

1.2 Construct the truth table for the statement “ $\text{not } (A \text{ and } (\text{not } B))$,” and use it to show that this statement is equivalent to the statement “ $A \Rightarrow B$.”

1.3 Prove DeMorgan’s law by constructing the appropriate truth tables.

1.4 Prove that for any statements A and B , we have “ $A \Leftrightarrow (A \text{ and } B)$ or $(A \text{ and } (\text{not } B))$.” This is useful because it allows us to prove a statement A by proving the two separate cases “ $(A \text{ and } B)$ ” and “ $(A \text{ and } (\text{not } B))$.” For example, to prove that $|x| \geq x$ for any $x \in \mathbb{R}$, we separately prove the cases “ $|x| \geq x$ and $x \geq 0$ ” and “ $|x| \geq x$ and $x < 0$.” Proving the two cases turns out to be easier than proving the statement $|x| \geq x$ directly (see Section 2.4 and Exercise 2.7).

1.5 (This exercise is adopted from [22, pp. 80–81]) Suppose that you are shown four cards, laid out in a row. Each card has a letter on one side and a number on the other. On the visible side of the cards are printed the symbols

$S \quad 8 \quad 3 \quad A$

Determine which cards you should turn over to decide if the following rule is true or false: “If there is a vowel on one side of the card, then there is an even number on the other side.”

CHAPTER 2

VECTOR SPACES AND MATRICES

2.1 Vector and Matrix

We define a *column n*-vector to be an array of n numbers, denoted

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}.$$

The number a_i is called the i th component of the vector \mathbf{a} . Denote by \mathbb{R} the set of real numbers and by \mathbb{R}^n the set of column n -vectors with real components. We call \mathbb{R}^n an n -dimensional *real vector space*. We commonly denote elements of \mathbb{R}^n by lowercase bold letters (e.g. \mathbf{x}). The components of $\mathbf{x} \in \mathbb{R}^n$ are denoted x_1, \dots, x_n .

We define a *row n*-vector as

$$[a_1, a_2, \dots, a_n].$$

The *transpose* of a given column vector \mathbf{a} is a row vector with corresponding elements, denoted \mathbf{a}^\top . For example, if

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix},$$

then

$$\mathbf{a}^\top = [a_1, a_2, \dots, a_n].$$

Equivalently, we may write $\underline{\mathbf{a}} = [a_1, a_2, \dots, a_n]^\top$. Throughout the text we adopt the convention that the term *vector* (without the qualifier *row* or *column*) refers to a column vector.

Two vectors $\mathbf{a} = [a_1, a_2, \dots, a_n]^\top$ and $\mathbf{b} = [b_1, b_2, \dots, b_n]^\top$ are equal if $a_i = b_i$, $i = 1, 2, \dots, n$.

The sum of the vectors \mathbf{a} and \mathbf{b} , denoted $\mathbf{a} + \mathbf{b}$, is the vector

$$\mathbf{a} + \mathbf{b} = [a_1 + b_1, a_2 + b_2, \dots, a_n + b_n]^\top.$$

The operation of addition of vectors has the following properties:

1. The operation is commutative:

$$\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a}.$$

2. The operation is associative:

$$(\mathbf{a} + \mathbf{b}) + \mathbf{c} = \mathbf{a} + (\mathbf{b} + \mathbf{c}).$$

3. There is a zero vector

$$\mathbf{0} = [0, 0, \dots, 0]^\top$$

such that

$$\mathbf{a} + \mathbf{0} = \mathbf{0} + \mathbf{a} = \mathbf{a}.$$

The vector

$$[a_1 - b_1, a_2 - b_2, \dots, a_n - b_n]^\top$$

is called the difference between \mathbf{a} and \mathbf{b} and is denoted $\mathbf{a} - \mathbf{b}$.

The vector $\mathbf{0} - \mathbf{b}$ is denoted $-\mathbf{b}$. Note that

$$\begin{aligned} \mathbf{b} + (\mathbf{a} - \mathbf{b}) &= \mathbf{a}, \\ -(-\mathbf{b}) &= \mathbf{b}, \\ -(\mathbf{a} - \mathbf{b}) &= \mathbf{b} - \mathbf{a}. \end{aligned}$$

The vector $\mathbf{b} - \mathbf{a}$ is the unique solution of the vector equation

$$\boxed{\mathbf{a} + \mathbf{x} = \mathbf{b}.}$$

Indeed, suppose that $\mathbf{x} = [x_1, x_2, \dots, x_n]^\top$ is a solution to $\mathbf{a} + \mathbf{x} = \mathbf{b}$. Then,

$$a_1 + x_1 = b_1,$$

$$a_2 + x_2 = b_2,$$

$$\vdots$$

$$a_n + x_n = b_n,$$

and thus

$$\boxed{\mathbf{x} = \mathbf{b} - \mathbf{a}.}$$

We define an operation of multiplication of a vector $\mathbf{a} \in \mathbb{R}^n$ by a real scalar $\alpha \in \mathbb{R}$ as

$$\alpha\mathbf{a} = [\alpha a_1, \alpha a_2, \dots, \alpha a_n]^\top.$$

This operation has the following properties:

1. The operation is distributive: for any real scalars α and β ,

$$\alpha(\mathbf{a} + \mathbf{b}) = \alpha\mathbf{a} + \alpha\mathbf{b},$$

$$(\alpha + \beta)\mathbf{a} = \alpha\mathbf{a} + \beta\mathbf{a}.$$

2. The operation is associative:

$$\alpha(\beta\mathbf{a}) = (\alpha\beta)\mathbf{a}.$$

3. The scalar 1 satisfies

$$1\mathbf{a} = \mathbf{a}.$$

4. Any scalar α satisfies

$$\alpha\mathbf{0} = \mathbf{0}.$$

5. The scalar 0 satisfies

$$0\mathbf{a} = \underline{0} \stackrel{=}{=} \mathbf{0} \quad \begin{matrix} 0 \in \mathbb{R} \\ \mathbf{0} \in \mathbb{R}^n \end{matrix}$$

6. The scalar -1 satisfies

$$(-1)\mathbf{a} = -\mathbf{a}.$$

Note that $\alpha\mathbf{a} = \mathbf{0}$ if and only if $\alpha = 0$ or $\mathbf{a} = \mathbf{0}$. To see this, observe that $\alpha\mathbf{a} = \mathbf{0}$ is equivalent to $\alpha a_1 = \alpha a_2 = \dots = \alpha a_n = 0$. If $\alpha = 0$ or $\mathbf{a} = \mathbf{0}$, then $\alpha\mathbf{a} = \mathbf{0}$. If $\mathbf{a} \neq \mathbf{0}$, then at least one of its components $a_k \neq 0$. For this component, $\alpha a_k = 0$, and hence we must have $\alpha = 0$. Similar arguments can be applied to the case when $\alpha \neq 0$.

A set of vectors $\{a_1, \dots, a_k\}$ is said to be *linearly independent* if the equality

$$\alpha_1 a_1 + \alpha_2 a_2 + \cdots + \alpha_k a_k = \mathbf{0} \quad \alpha_i = 0$$

implies that all coefficients α_i , $i = 1, \dots, k$, are equal to zero. A set of the vectors $\{a_1, \dots, a_k\}$ is *linearly dependent* if it is not linearly independent.

Note that the set composed of the single vector $\mathbf{0}$ is linearly dependent, for if $\alpha \neq 0$, then $\alpha \mathbf{0} = \mathbf{0}$. In fact, any set of vectors containing the vector $\mathbf{0}$ is linearly dependent.

A set composed of a single nonzero vector $a \neq \mathbf{0}$ is linearly independent since $\alpha a = \mathbf{0}$ implies that $\alpha = 0$.

A vector a is said to be a *linear combination* of vectors a_1, a_2, \dots, a_k if there are scalars $\alpha_1, \dots, \alpha_k$ such that

$$a = \alpha_1 a_1 + \alpha_2 a_2 + \cdots + \alpha_k a_k.$$

Proposition 2.1 A set of vectors $\{a_1, a_2, \dots, a_k\}$ is linearly dependent if and only if one of the vectors from the set is a linear combination of the remaining vectors. \square

Proof. \Rightarrow : If $\{a_1, a_2, \dots, a_k\}$ is linearly dependent, then

$$\alpha_1 a_1 + \alpha_2 a_2 + \cdots + \alpha_k a_k = \mathbf{0},$$

where at least one of the scalars $\alpha_i \neq 0$, whence

$$a_i = -\frac{\alpha_1}{\alpha_i} a_1 - \frac{\alpha_2}{\alpha_i} a_2 - \cdots - \frac{\alpha_k}{\alpha_i} a_k.$$

\Leftarrow : Suppose that

$$a_1 = \alpha_2 a_2 + \alpha_3 a_3 + \cdots + \alpha_k a_k,$$

then

$$(-1)a_1 + \alpha_2 a_2 + \cdots + \alpha_k a_k = \mathbf{0}.$$

Because the first scalar is nonzero, the set of vectors $\{a_1, a_2, \dots, a_k\}$ is linearly dependent. The same argument holds if a_i , $i = 2, \dots, k$, is a linear combination of the remaining vectors. \blacksquare

→ A subset \mathcal{V} of \mathbb{R}^n is called a *subspace* of \mathbb{R}^n if \mathcal{V} is closed under the operations of vector addition and scalar multiplication. That is, if a and b are vectors in \mathcal{V} , then the vectors $a + b$ and αa are also in \mathcal{V} for every scalar α .

Every subspace contains the zero vector $\mathbf{0}$ for if a is an element of the subspace, so is $(-1)a = -a$. Hence, $a - a = \mathbf{0}$ also belongs to the subspace.

Let a_1, a_2, \dots, a_k be arbitrary vectors in \mathbb{R}^n . The set of all their linear combinations is called the *span* of a_1, a_2, \dots, a_k and is denoted

$$\text{span}[a_1, a_2, \dots, a_k] = \left\{ \sum_{i=1}^k \alpha_i a_i : \alpha_1, \dots, \alpha_k \in \mathbb{R} \right\}.$$

Given a vector \mathbf{a} , the subspace $\text{span}[\mathbf{a}]$ is composed of the vectors $\alpha\mathbf{a}$, where α is an arbitrary real number ($\alpha \in \mathbb{R}$). Also observe that if \mathbf{a} is a linear combination of $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k$, then

$$\text{span}[\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k, \mathbf{a}] = \text{span}[\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k].$$

$$\dim \mathcal{V} = k$$

The span of any set of vectors is a subspace.

Given a subspace \mathcal{V} , any set of linearly independent vectors $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\} \subset \mathcal{V}$ such that $\mathcal{V} = \text{span}[\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k]$ is referred to as a basis of the subspace \mathcal{V} . All bases of a subspace \mathcal{V} contain the same number of vectors. This number is called the dimension of \mathcal{V} , denoted $\dim \mathcal{V}$.

Proposition 2.2 If $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\}$ is a basis of \mathcal{V} , then any vector \mathbf{a} of \mathcal{V} can be represented uniquely as

$$\mathbf{a} = \alpha_1 \mathbf{a}_1 + \alpha_2 \mathbf{a}_2 + \cdots + \alpha_k \mathbf{a}_k,$$

where $\alpha_i \in \mathbb{R}$, $i = 1, 2, \dots, k$.

□

Proof. To prove the uniqueness of the representation of \mathbf{a} in terms of the basis vectors, assume that

$$\mathbf{a} = \alpha_1 \mathbf{a}_1 + \alpha_2 \mathbf{a}_2 + \cdots + \alpha_k \mathbf{a}_k$$

and

$$\mathbf{a} = \beta_1 \mathbf{a}_1 + \beta_2 \mathbf{a}_2 + \cdots + \beta_k \mathbf{a}_k.$$

We now show that $\alpha_i = \beta_i$, $i = 1, \dots, k$. We have

$$\alpha_1 \mathbf{a}_1 + \alpha_2 \mathbf{a}_2 + \cdots + \alpha_k \mathbf{a}_k = \beta_1 \mathbf{a}_1 + \beta_2 \mathbf{a}_2 + \cdots + \beta_k \mathbf{a}_k$$

or

$$(\alpha_1 - \beta_1) \mathbf{a}_1 + (\alpha_2 - \beta_2) \mathbf{a}_2 + \cdots + (\alpha_k - \beta_k) \mathbf{a}_k = \mathbf{0}.$$

Because the set $\{\mathbf{a}_i : i = 1, 2, \dots, k\}$ is linearly independent, $\alpha_1 - \beta_1 = \alpha_2 - \beta_2 = \cdots = \alpha_k - \beta_k = 0$, which implies that $\alpha_i = \beta_i$, $i = 1, \dots, k$. ■

Suppose that we are given a basis $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\}$ of \mathcal{V} and a vector $\mathbf{a} \in \mathcal{V}$ such that

$$\mathbf{a} = \alpha_1 \mathbf{a}_1 + \alpha_2 \mathbf{a}_2 + \cdots + \alpha_k \mathbf{a}_k.$$

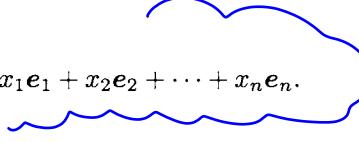
The coefficients α_i , $i = 1, \dots, k$, are called the *coordinates* of \mathbf{a} with respect to the basis $\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k\}$.

The *natural basis* for \mathbb{R}^n is the set of vectors

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad \dots, \quad e_n = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

$$\mathcal{V} \subseteq \mathbb{R}^n$$

The reason for calling these vectors the natural basis is that

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2 + \cdots + x_n \mathbf{e}_n.$$


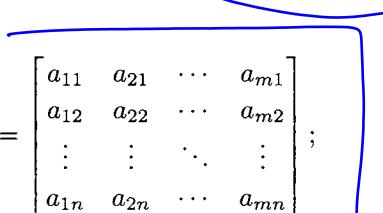
We can similarly define *complex vector spaces*. For this, let \mathbb{C} denote the set of complex numbers and \mathbb{C}^n the set of column n -vectors with complex components. As the reader can easily verify, the set \mathbb{C}^n has properties similar to those of \mathbb{R}^n , where scalars can take complex values.

A *matrix* is a rectangular array of numbers, commonly denoted by upper-case bold letters (e.g., \mathbf{A}). A matrix with m rows and n columns is called an $m \times n$ matrix, and we write

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

The real number a_{ij} located in the i th row and j th column is called the (i, j) th *entry*. We can think of \mathbf{A} in terms of its n columns, each of which is a column vector in \mathbb{R}^m . Alternatively, we can think of \mathbf{A} in terms of its m rows, each of which is a row n -vector.

Consider the $m \times n$ matrix \mathbf{A} above. The *transpose* of matrix \mathbf{A} , denoted \mathbf{A}^\top , is the $n \times m$ matrix

$$\mathbf{A}^\top = \begin{bmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix};$$


that is, the columns of \mathbf{A} are the rows of \mathbf{A}^\top , and vice versa.

Let the symbol $\mathbb{R}^{m \times n}$ denote the set of $m \times n$ matrices whose entries are real numbers. We treat column vectors in \mathbb{R}^n as elements of $\mathbb{R}^{n \times 1}$. Similarly, we treat row n -vectors as elements of $\mathbb{R}^{1 \times n}$. Accordingly, vector transposition is simply a special case of matrix transposition, and we will no longer distinguish between the two. Note that there is a slight inconsistency in the notation of row vectors when identified as $1 \times n$ matrices: We separate the components of the row vector with commas, whereas in matrix notation we do not generally use commas. However, the use of commas in separating elements in a row helps to clarify their separation. We use such commas even in separating matrices arranged in a horizontal row.

2.2 Rank of a Matrix

Consider the $m \times n$ matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \cdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}.$$

Let us denote the k th column of \mathbf{A} by \mathbf{a}_k :

$$\mathbf{a}_k = \begin{bmatrix} a_{1k} \\ a_{2k} \\ \vdots \\ a_{mk} \end{bmatrix}.$$

*جواب: وجہ کیمی
! جبکہ وجہ نہیں
Rank(A)*

The maximal number of linearly independent columns of \mathbf{A} is called the *rank* of the matrix \mathbf{A} , denoted $\text{rank } \mathbf{A}$. Note that $\text{rank } \mathbf{A}$ is the dimension of $\text{span}[\mathbf{a}_1, \dots, \mathbf{a}_n]$.

Proposition 2.3 *The rank of a matrix \mathbf{A} is invariant under the following operations:*

- 1. Multiplication of the columns of \mathbf{A} by nonzero scalars.
- 2. Interchange of the columns.
- 3. Addition to a given column a linear combination of other columns. \square

Proof.

1. Let $\mathbf{b}_k = \alpha_k \mathbf{a}_k$, where $\alpha_k \neq 0$, $k = 1, \dots, n$, and let $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n]$. Obviously,

$$\text{span}[\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] = \text{span}[\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n],$$

and thus

$$\text{rank } \mathbf{A} = \text{rank } \mathbf{B}.$$

2. The number of linearly independent vectors does not depend on their order.

3. Let

$$\mathbf{b}_1 = \mathbf{a}_1 + c_2 \mathbf{a}_2 + \cdots + c_n \mathbf{a}_n,$$

$$\mathbf{b}_2 = \mathbf{a}_2,$$

⋮

$$\mathbf{b}_n = \mathbf{a}_n.$$

So, for any $\alpha_1, \dots, \alpha_n$,

$$\alpha_1 \mathbf{b}_1 + \alpha_2 \mathbf{b}_2 + \cdots + \alpha_n \mathbf{b}_n = \alpha_1 \mathbf{a}_1 + (\alpha_2 + \alpha_1 c_2) \mathbf{a}_2 + \cdots + (\alpha_n + \alpha_1 c_n) \mathbf{a}_n,$$

and hence

$$\text{span}[\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n] \subset \text{span}[\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n].$$

On the other hand,

$$\begin{aligned}\mathbf{a}_1 &= \mathbf{b}_1 - c_2 \mathbf{b}_2 - \cdots - c_n \mathbf{b}_n, \\ \mathbf{a}_2 &= \mathbf{b}_2, \\ &\vdots \\ \mathbf{a}_n &= \mathbf{b}_n.\end{aligned}$$

Hence,

$$\text{span}[\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n] \subset \text{span}[\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n].$$

Therefore, $\text{rank } \mathbf{A} = \text{rank } \mathbf{B}$. ■

A matrix \mathbf{A} is said to be *square* if the number of its rows is equal to the number of its columns (i.e., it is $n \times n$). Associated with each square matrix \mathbf{A} is a scalar called the *determinant* of the matrix \mathbf{A} , denoted $\det \mathbf{A}$ or $| \mathbf{A} |$. The determinant of a square matrix is a function of its columns and has the following properties:

1. The determinant of the matrix $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n]$ is a linear function of each column; that is,

$$\begin{aligned}\det[\mathbf{a}_1, \dots, \mathbf{a}_{k-1}, \alpha \mathbf{a}_k^{(1)} + \beta \mathbf{a}_k^{(2)}, \mathbf{a}_{k+1}, \dots, \mathbf{a}_n] \\ = \alpha \det[\mathbf{a}_1, \dots, \mathbf{a}_{k-1}, \mathbf{a}_k^{(1)}, \mathbf{a}_{k+1}, \dots, \mathbf{a}_n] \\ + \beta \det[\mathbf{a}_1, \dots, \mathbf{a}_{k-1}, \mathbf{a}_k^{(2)}, \mathbf{a}_{k+1}, \dots, \mathbf{a}_n]\end{aligned}$$

\det
 $| \mathbf{A} |$

for each $\alpha, \beta \in \mathbb{R}$, $\mathbf{a}_k^{(1)}, \mathbf{a}_k^{(2)} \in \mathbb{R}^n$.

2. If for some k we have $\mathbf{a}_k = \mathbf{a}_{k+1}$, then

$$\det \mathbf{A} = \det[\mathbf{a}_1, \dots, \mathbf{a}_k, \mathbf{a}_{k+1}, \dots, \mathbf{a}_n] = \det[\mathbf{a}_1, \dots, \mathbf{a}_k, \mathbf{a}_k, \dots, \mathbf{a}_n] = 0.$$

$\det: \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$

3. Let

$$\mathbf{I}_n = [\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n] = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix},$$

where $\{e_1, \dots, e_n\}$ is the natural basis for \mathbb{R}^n . Then

$$\det I_n = 1.$$

Note that if $\alpha = \beta = 0$ in property 1, then

$$\det[a_1, \dots, a_{k-1}, \underline{\mathbf{0}}, \underline{a_{k+1}, \dots, a_n}] = 0.$$

Thus, if one of the columns is $\mathbf{0}$, then the determinant is equal to zero.

The determinant does not change its value if we add to a column another column multiplied by a scalar. This follows from properties 1 and 2 as shown below:

$$\begin{aligned} \det[a_1, \dots, a_{k-1}, a_k + \alpha a_j, a_{k+1}, \dots, a_j, \dots, a_n] \\ = \det[a_1, \dots, a_{k-1}, a_k, a_{k+1}, \dots, a_j, \dots, a_n] \\ + \alpha \det[a_1, \dots, a_{k-1}, a_j, a_{k+1}, \dots, a_j, \dots, a_n] \\ = \det[a_1, \dots, a_n]. \end{aligned}$$

However, the determinant changes its sign if we interchange columns. To show this property, note that

$$\begin{aligned} \det[a_1, \dots, a_{k-1}, a_k, a_{k+1}, \dots, a_n] \\ = \det[a_1, \dots, a_k + a_{k+1}, a_{k+1}, \dots, a_n] \\ = \det[a_1, \dots, a_k + a_{k+1}, a_{k+1} - (a_k + a_{k+1}), \dots, a_n] \\ = \det[a_1, \dots, a_k + a_{k+1}, -a_k, \dots, a_n] \\ = -\det[a_1, \dots, a_k + a_{k+1}, a_k, \dots, a_n] \\ = -(\det[a_1, \dots, a_k, a_k, \dots, a_n] + \det[a_1, \dots, a_{k+1}, a_k, \dots, a_n]) \\ = -\det[a_1, \dots, a_{k+1}, a_k, \dots, a_n]. \end{aligned}$$

A p th-order *minor* of an $m \times n$ matrix \mathbf{A} , with $p \leq \min\{m, n\}$, is the determinant of a $p \times p$ matrix obtained from \mathbf{A} by deleting $m - p$ rows and $n - p$ columns. (The notation $\min\{m, n\}$ represents the smaller of m and n .)

We can use minors to investigate the rank of a matrix. In particular, we have the following proposition.

Proposition 2.4 *If an $m \times n$ ($m \geq n$) matrix \mathbf{A} has a nonzero n th-order minor, then the columns of \mathbf{A} are linearly independent; that is, $\text{rank } \mathbf{A} = n$.*

□

Proof. Suppose that \mathbf{A} has a nonzero n th-order minor. Without loss of generality, we assume that the n th-order minor corresponding to the first n rows of \mathbf{A} is nonzero. Let x_i , $i = 1, \dots, n$, be scalars such that

$$x_1 a_1 + x_2 a_2 + \dots + x_n a_n = \mathbf{0}.$$

$$\text{Rank} \left[\begin{array}{cccc|c} 0 & 0 & 0 & 0 & \cdot \\ 0 & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & \cdot \\ \vdots & & & & \vdots \end{array} \right] = n$$

$\det(\cdot) \neq 0$

The vector equality above is equivalent to the following set of m equations:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= 0 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= 0 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= 0 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= 0. \end{aligned}$$

For $i = 1, \dots, n$, let

$$\tilde{\mathbf{a}}_i = \begin{bmatrix} a_{1i} \\ \vdots \\ a_{ni} \end{bmatrix}.$$

Then, $x_1\tilde{\mathbf{a}}_1 + \cdots + x_n\tilde{\mathbf{a}}_n = \mathbf{0}$.

The n th-order minor is $\det[\tilde{\mathbf{a}}_1, \tilde{\mathbf{a}}_2, \dots, \tilde{\mathbf{a}}_n]$, assumed to be nonzero. From the properties of determinants it follows that the columns $\tilde{\mathbf{a}}_1, \tilde{\mathbf{a}}_2, \dots, \tilde{\mathbf{a}}_n$ are linearly independent. Therefore, all $x_i = 0$, $i = 1, \dots, n$. Hence, the columns $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$ are linearly independent. ■

From the above it follows that if there is a nonzero minor, then the columns associated with this nonzero minor are linearly independent.

If a matrix \mathbf{A} has an r th-order minor $|\mathbf{M}|$ with the properties (i) $|\mathbf{M}| \neq 0$ and (ii) any minor of \mathbf{A} that is formed by adding a row and a column of \mathbf{A} to \mathbf{M} is zero, then

$$\text{rank } \mathbf{A} = r.$$

Thus, the rank of a matrix is equal to the highest order of its nonzero minor(s).

A *nonsingular* (or *invertible*) matrix is a square matrix whose determinant is nonzero. Suppose that \mathbf{A} is an $n \times n$ square matrix. Then, \mathbf{A} is nonsingular if and only if there is another $n \times n$ matrix $\underline{\mathbf{B}}$ such that

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}_n,$$

where \mathbf{I}_n denotes the $n \times n$ *identity matrix*:

$$\mathbf{I}_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

We call the matrix \mathbf{B} above the inverse matrix of \mathbf{A} , and write $\mathbf{B} = \mathbf{A}^{-1}$.

2.3 Linear Equations

Suppose that we are given m equations in n unknowns of the form

$$\left. \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m. \end{array} \right\}$$

We can represent the set of equations above as a vector equation

$$x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 + \cdots + x_n \mathbf{a}_n = \mathbf{b},$$

where

$$\mathbf{a}_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

Associated with this system of equations is the matrix

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n],$$

and an augmented matrix

$$[\mathbf{A}, \mathbf{b}] = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n, \mathbf{b}].$$

We can also represent the system of equations above as

$$\mathbf{Ax} = \mathbf{b},$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

Theorem 2.1 *The system of equations $\mathbf{Ax} = \mathbf{b}$ has a solution if and only if*

$$\text{rank } \mathbf{A} = \text{rank}[\mathbf{A}, \mathbf{b}].$$

□

Proof. \Rightarrow : Suppose that the system $\mathbf{Ax} = \mathbf{b}$ has a solution. Therefore, \mathbf{b} is a linear combination of the columns of \mathbf{A} ; that is, there exist x_1, \dots, x_n such

that $x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_n\mathbf{a}_n = \mathbf{b}$. It follows that \mathbf{b} belongs to $\text{span}[\mathbf{a}_1, \dots, \mathbf{a}_n]$ and hence

$$\begin{aligned}\text{rank } \mathbf{A} &= \dim \text{span}[\mathbf{a}_1, \dots, \mathbf{a}_n] \\ &= \dim \text{span}[\mathbf{a}_1, \dots, \mathbf{a}_n, \mathbf{b}] \\ &= \text{rank}[\mathbf{A}, \mathbf{b}].\end{aligned}$$

\Leftarrow : Suppose that $\text{rank } \mathbf{A} = \text{rank}[\mathbf{A}, \mathbf{b}] = r$. Thus, we have r linearly independent columns of \mathbf{A} . Without loss of generality, let $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_r$ be these columns. Therefore, $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_r$ are also linearly independent columns of the matrix $[\mathbf{A}, \mathbf{b}]$. Because $\text{rank}[\mathbf{A}, \mathbf{b}] = r$, the remaining columns of $[\mathbf{A}, \mathbf{b}]$ can be expressed as linear combinations of $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_r$. In particular, \mathbf{b} can be expressed as a linear combination of these columns. Hence, there exist x_1, \dots, x_n such that $x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_n\mathbf{a}_n = \mathbf{b}$. ■



Theorem 2.2 Consider the equation $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\text{rank } \mathbf{A} = m$. A solution to $\mathbf{Ax} = \mathbf{b}$ can be obtained by assigning arbitrary values for $n - m$ variables and solving for the remaining ones. □

Proof. We have $\text{rank } \mathbf{A} = m$, and therefore we can find m linearly independent columns of \mathbf{A} . Without loss of generality, let $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m$ be such columns. Rewrite the equation $\mathbf{Ax} = \mathbf{b}$ as

$$x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + \cdots + x_m\mathbf{a}_m = \mathbf{b} - x_{m+1}\mathbf{a}_{m+1} - \cdots - x_n\mathbf{a}_n.$$

Assign to $x_{m+1}, x_{m+2}, \dots, x_n$ arbitrary values, say

$$x_{m+1} = d_{m+1}, x_{m+2} = d_{m+2}, \dots, x_n = d_n,$$

and let

$$\mathbf{B} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m] \in \mathbb{R}^{m \times m}.$$

Note that $\det \mathbf{B} \neq 0$. We can represent the system of equations above as

$$\mathbf{B} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = [\mathbf{b} - d_{m+1}\mathbf{a}_{m+1} - \cdots - d_n\mathbf{a}_n].$$

The matrix \mathbf{B} is invertible, and therefore we can solve for $[x_1, x_2, \dots, x_m]^\top$. Specifically,

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \mathbf{B}^{-1} [\mathbf{b} - d_{m+1}\mathbf{a}_{m+1} - \cdots - d_n\mathbf{a}_n].$$



2.4 Inner Products and Norms

The absolute value of a real number a , denoted $|a|$, is defined as

$$|a| = \begin{cases} a & \text{if } a \geq 0 \\ -a & \text{if } a < 0. \end{cases}$$

The following formulas hold:

1. $|a| = |-a|$.
2. $-|a| \leq a \leq |a|$.
3. $|a + b| \leq |a| + |b|$.
4. $||a| - |b|| \leq |a - b| \leq |a| + |b|$.
5. $|ab| = |a||b|$.
6. $|a| \leq c$ and $|b| \leq d$ imply that $|a + b| \leq c + d$.
7. The inequality $|a| < b$ is equivalent to $-b < a < b$ (i.e., $a < b$ and $-a < b$). The same holds if we replace every occurrence of " $<$ " by " \leq ".
8. The inequality $|a| > b$ is equivalent to $a > b$ or $-a > b$. The same holds if we replace every occurrence of " $>$ " by " \geq ".

For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we define the *Euclidean inner product* by

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i = \mathbf{x}^\top \mathbf{y}.$$

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^n x_i y_i$$

The inner product is a real-valued function $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ having the following properties:

1. Positivity: $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$, $\langle \mathbf{x}, \mathbf{x} \rangle = 0$ if and only if $\mathbf{x} = \mathbf{0}$.
2. Symmetry: $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle$.
3. Additivity: $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle$.
4. Homogeneity: $\langle r\mathbf{x}, \mathbf{y} \rangle = r\langle \mathbf{x}, \mathbf{y} \rangle$ for every $r \in \mathbb{R}$.

The properties of additivity and homogeneity in the second vector also hold; that is,

$$\begin{aligned} \langle \mathbf{x}, \mathbf{y} + \mathbf{z} \rangle &= \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{z} \rangle, \\ \langle \mathbf{x}, r\mathbf{y} \rangle &= r\langle \mathbf{x}, \mathbf{y} \rangle \quad \text{for every } r \in \mathbb{R}. \end{aligned}$$

The above can be shown using properties 2 to 4. Indeed,

$$\begin{aligned}\langle \mathbf{x}, \mathbf{y} + \mathbf{z} \rangle &= \langle \mathbf{y} + \mathbf{z}, \mathbf{x} \rangle \\ &= \langle \mathbf{y}, \mathbf{x} \rangle + \langle \mathbf{z}, \mathbf{x} \rangle \\ &= \langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x}, \mathbf{z} \rangle\end{aligned}$$

and

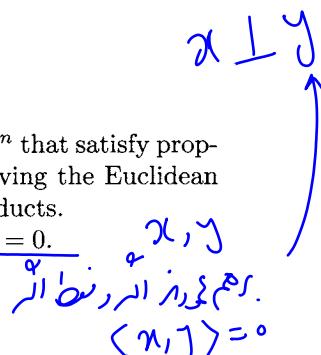
$$\langle \mathbf{x}, r\mathbf{y} \rangle = \langle r\mathbf{y}, \mathbf{x} \rangle = r\langle \mathbf{y}, \mathbf{x} \rangle = r\langle \mathbf{x}, \mathbf{y} \rangle.$$

It is possible to define other real-valued functions on $\mathbb{R}^n \times \mathbb{R}^n$ that satisfy properties 1 to 4 above (see Exercise 2.8). Many results involving the Euclidean inner product also hold for these other forms of inner products.

The vectors \mathbf{x} and \mathbf{y} are said to be orthogonal if $\langle \mathbf{x}, \mathbf{y} \rangle = 0$.

The *Euclidean norm* of a vector \mathbf{x} is defined as

$$\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \sqrt{\mathbf{x}^\top \mathbf{x}}.$$



Theorem 2.3 Cauchy-Schwarz Inequality. For any two vectors \mathbf{x} and \mathbf{y} in \mathbb{R}^n , the Cauchy-Schwarz inequality

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\| \|\mathbf{y}\|$$

holds. Furthermore, equality holds if and only if $\mathbf{x} = \alpha\mathbf{y}$ for some $\alpha \in \mathbb{R}$. \square

Proof. First assume that \mathbf{x} and \mathbf{y} are unit vectors; that is, $\|\mathbf{x}\| = \|\mathbf{y}\| = 1$. Then,

$$\begin{aligned}0 \leq \|\mathbf{x} - \mathbf{y}\|^2 &= \langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle \\ &= \|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \mathbf{y} \rangle + \|\mathbf{y}\|^2 \\ &= 2 - 2\langle \mathbf{x}, \mathbf{y} \rangle\end{aligned}$$

or

$$\langle \mathbf{x}, \mathbf{y} \rangle \leq 1,$$

with equality holding if and only if $\mathbf{x} = \mathbf{y}$.

Next, assuming that neither \mathbf{x} nor \mathbf{y} is zero (for the inequality obviously holds if one of them is zero), we replace \mathbf{x} and \mathbf{y} by the unit vectors $\mathbf{x}/\|\mathbf{x}\|$ and $\mathbf{y}/\|\mathbf{y}\|$. Then, apply property 4 to get

$$\langle \mathbf{x}, \mathbf{y} \rangle \leq \|\mathbf{x}\| \|\mathbf{y}\|.$$

Now replace \mathbf{x} by $-\mathbf{x}$ and again apply property 4 to get

$$-\langle \mathbf{x}, \mathbf{y} \rangle \leq \|\mathbf{x}\| \|\mathbf{y}\|.$$

The last two inequalities imply the absolute value inequality. Equality holds if and only if $\mathbf{x}/\|\mathbf{x}\| = \pm \mathbf{y}/\|\mathbf{y}\|$; that is, $\mathbf{x} = \alpha\mathbf{y}$ for some $\alpha \in \mathbb{R}$. \blacksquare

The Euclidean norm of a vector $\|\mathbf{x}\|$ has the following properties:

1. Positivity: $\|\mathbf{x}\| \geq 0$, $\|\mathbf{x}\| = 0$ if and only if $\mathbf{x} = \mathbf{0}$.
2. Homogeneity: $\|r\mathbf{x}\| = |r|\|\mathbf{x}\|$, $r \in \mathbb{R}$.
3. Triangle inequality: $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$.

The triangle inequality can be proved using the Cauchy-Schwarz inequality, as follows. We have

$$\|\mathbf{x} + \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + 2\langle \mathbf{x}, \mathbf{y} \rangle + \|\mathbf{y}\|^2.$$

By the Cauchy-Schwarz inequality,

$$\begin{aligned} \|\mathbf{x} + \mathbf{y}\|^2 &\leq \|\mathbf{x}\|^2 + 2\|\mathbf{x}\|\|\mathbf{y}\| + \|\mathbf{y}\|^2 \\ &= (\|\mathbf{x}\| + \|\mathbf{y}\|)^2, \end{aligned}$$

and therefore

$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|.$$

Note that if \mathbf{x} and \mathbf{y} are orthogonal: $\langle \mathbf{x}, \mathbf{y} \rangle = 0$, then

$$\|\mathbf{x} + \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2,$$

$\leftarrow \mathbf{x} \perp \mathbf{y}$

which is the *Pythagorean theorem* for \mathbb{R}^n .

The Euclidean norm is an example of a general *vector norm*, which is any function satisfying the three properties of positivity, homogeneity, and triangle inequality. Other examples of vector norms on \mathbb{R}^n include the 1-norm, defined by $\|\mathbf{x}\|_1 = |x_1| + \cdots + |x_n|$, and the ∞ -norm, defined by $\|\mathbf{x}\|_\infty = \max_i |x_i|$ (where the notation \max_i represents the largest over all the possible index values of i). The Euclidean norm is often referred to as the *2-norm*, and denoted $\|\mathbf{x}\|_2$. The norms above are special cases of the *p-norm*, given by

$$\|\mathbf{x}\|_p = \begin{cases} (|x_1|^p + \cdots + |x_n|^p)^{1/p} & \text{if } 1 \leq p < \infty \\ \max\{|x_1|, \dots, |x_n|\} & \text{if } p = \infty. \end{cases}$$

We can use norms to define the notion of a continuous function, as follows. A function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is *continuous* at \mathbf{x} if for all $\varepsilon > 0$, there exists $\delta > 0$ such that $\|\mathbf{y} - \mathbf{x}\| < \delta \Rightarrow \|\mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{x})\| < \varepsilon$. If the function \mathbf{f} is continuous at every point in \mathbb{R}^n , we say that it is continuous on \mathbb{R}^n . Note that $\mathbf{f} = [f_1, \dots, f_m]^\top$ is continuous if and only if each component f_i , $i = 1, \dots, m$, is continuous.

For the complex vector space \mathbb{C}^n , we define an inner product $\langle \mathbf{x}, \mathbf{y} \rangle$ to be $\sum_{i=1}^n x_i \bar{y}_i$, where the bar denotes complex conjugation. The inner product on \mathbb{C}^n is a complex-valued function having the following properties:

1. $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0$, $\langle \mathbf{x}, \mathbf{x} \rangle = 0$ if and only if $\mathbf{x} = \mathbf{0}$.

2. $\langle \mathbf{x}, \mathbf{y} \rangle = \overline{\langle \mathbf{y}, \mathbf{x} \rangle}.$

3. $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle.$

4. $\langle r\mathbf{x}, \mathbf{y} \rangle = r\langle \mathbf{x}, \mathbf{y} \rangle,$ where $r \in \mathbb{C}.$

From properties 1 to 4, we can deduce other properties, such as

$$\langle \mathbf{x}, r_1\mathbf{y} + r_2\mathbf{z} \rangle = \bar{r}_1\langle \mathbf{x}, \mathbf{y} \rangle + \bar{r}_2\langle \mathbf{x}, \mathbf{z} \rangle,$$

where $r_1, r_2 \in \mathbb{C}.$ For $\mathbb{C}^n,$ the vector norm can similarly be defined by $\|\mathbf{x}\|^2 = \langle \mathbf{x}, \mathbf{x} \rangle.$ For more information, consult Gel'fand [47].

EXERCISES

2.1 Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\text{rank } \mathbf{A} = m.$ Show that $m \leq n.$

2.2 Prove that the system $\mathbf{Ax} = \mathbf{b}, \mathbf{A} \in \mathbb{R}^{m \times n},$ has a unique solution if and only if $\text{rank } \mathbf{A} = \text{rank}[\mathbf{A}, \mathbf{b}] = n.$

2.3 (Adapted from [38].) We know that if $k \geq n + 1,$ then the vectors $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_k \in \mathbb{R}^n$ are linearly dependent; that is, there exist scalars $\alpha_1, \dots, \alpha_k$ such that at least one $\alpha_i \neq 0$ and $\sum_{i=1}^k \alpha_i \mathbf{a}_i = \mathbf{0}.$ Show that if $k \geq n + 2,$ then there exist scalars $\alpha_1, \dots, \alpha_k$ such that at least one $\alpha_i \neq 0,$ $\sum_{i=1}^k \alpha_i \mathbf{a}_i = \mathbf{0},$ and $\sum_{i=1}^k \alpha_i = 0.$

Hint: Introduce the vectors $\bar{\mathbf{a}}_i = [1, \mathbf{a}_i^\top]^\top \in \mathbb{R}^{n+1}, i = 1, \dots, k,$ and use the fact that any $n + 2$ vectors in \mathbb{R}^{n+1} are linearly dependent.

2.4 Consider an $m \times m$ matrix \mathbf{M} that has block form

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_{m-k,k} & \mathbf{I}_{m-k} \\ \mathbf{M}_{k,k} & \mathbf{O}_{k,m-k} \end{bmatrix},$$

where $\mathbf{M}_{k,k}$ is $k \times k,$ $\mathbf{M}_{m-k,k}$ is $(m - k) \times k,$ \mathbf{I}_{m-k} is the $(m - k) \times (m - k)$ identity matrix, and $\mathbf{O}_{k,m-k}$ is the $k \times (m - k)$ zero matrix.

a. Show that

$$|\det \mathbf{M}| = |\det \mathbf{M}_{k,k}|.$$

This result is relevant to the proof of Proposition 19.1.

b. Under certain assumptions, the following stronger result holds:

$$\det \mathbf{M} = \det(-\mathbf{M}_{k,k})$$

Identify cases where this is true, and show that it is false in general.

Übung

- 2.5** It is well known that for any $a, b, c, d \in \mathbb{C}$,

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc.$$

Suppose now that \mathbf{A} , \mathbf{B} , \mathbf{C} , and \mathbf{D} are real or complex square matrices of the same size. Give a sufficient condition under which

$$\det \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} = \mathbf{AD} - \mathbf{BC}.$$

An interesting discussion on determinants of block matrices is provided in [121].

- 2.6** Consider the following system of linear equations:

$$\begin{aligned} x_1 + x_2 + 2x_3 + x_4 &= 1 \\ x_1 - 2x_2 - x_4 &= -2. \end{aligned}$$

Use Theorem 2.1 to check if the system has a solution. Then, use the method of Theorem 2.2 to find a general solution to the system.

- 2.7** Prove the seven properties of the absolute value of a real number.

- 2.8** Consider the function $\langle \cdot, \cdot \rangle_2 : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow \mathbb{R}$, defined by $\langle \mathbf{x}, \mathbf{y} \rangle_2 = 2x_1y_1 + 3x_2y_1 + 3x_1y_2 + 5x_2y_2$, where $\mathbf{x} = [x_1, x_2]^\top$ and $\mathbf{y} = [y_1, y_2]^\top$. Show that $\langle \cdot, \cdot \rangle_2$ satisfies conditions 1 to 4 for inner products.

Note: This is a special case of Exercise 3.21.

- 2.9** Show that for any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $|||\mathbf{x}| - |\mathbf{y}||| \leq \|\mathbf{x} - \mathbf{y}\|$.

Hint: Write $\mathbf{x} = (\mathbf{x} - \mathbf{y}) + \mathbf{y}$, and use the triangle inequality. Do the same for \mathbf{y} .

- 2.10** Use Exercise 2.9 to show that the norm $\|\cdot\|$ is a *uniformly continuous function*; that is, for all $\varepsilon > 0$, there exists $\delta > 0$ such that if $\|\mathbf{x} - \mathbf{y}\| < \delta$, then $|||\mathbf{x}| - |\mathbf{y}||| < \varepsilon$.

! Gezeigt ist

CHAPTER 3

TRANSFORMATIONS

3.1 Linear Transformations

جُنْدَلِي

A function $\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is called a *linear transformation* if:

1. $\mathcal{L}(ax) = a\mathcal{L}(x)$ for every $x \in \mathbb{R}^n$ and $a \in \mathbb{R}$.
2. $\mathcal{L}(x + y) = \mathcal{L}(x) + \mathcal{L}(y)$ for every $x, y \in \mathbb{R}^n$.

If we fix the bases for \mathbb{R}^n and \mathbb{R}^m , then the linear transformation \mathcal{L} can be represented by a matrix. Specifically, there exists $A \in \mathbb{R}^{m \times n}$ such that the following representation holds. Suppose that $x \in \mathbb{R}^n$ is a given vector, and x' is the representation of x with respect to the given basis for \mathbb{R}^n . If $y = \mathcal{L}(x)$ and y' is the representation of y with respect to the given basis for \mathbb{R}^m , then

$$y' = Ax'.$$

$A \in \mathbb{R}^{m \times n}$

$y = Ax$

We call A the *matrix representation* of \mathcal{L} with respect to the given bases for \mathbb{R}^n and \mathbb{R}^m . In the special case where we assume the natural bases for \mathbb{R}^n and \mathbb{R}^m , the matrix representation A satisfies

$$\mathcal{L}(x) = Ax.$$

Let $\{e_1, e_2, \dots, e_n\}$ and $\{e'_1, e'_2, \dots, e'_n\}$ be two bases for \mathbb{R}^n . Define the matrix

$$T = [e'_1, e'_2, \dots, e'_n]^{-1} [e_1, e_2, \dots, e_n].$$

We call T the *transformation matrix* from $\{e_1, e_2, \dots, e_n\}$ to $\{e'_1, e'_2, \dots, e'_n\}$. It is clear that

$$[e_1, e_2, \dots, e_n] = [e'_1, e'_2, \dots, e'_n] T;$$

that is, the i th column of T is the vector of coordinates of e_i with respect to the basis $\{e'_1, e'_2, \dots, e'_n\}$.

Fix a vector in \mathbb{R}^n . Let x be the column of the coordinates of the vector with respect to $\{e_1, e_2, \dots, e_n\}$ and x' the coordinates of the same vector with respect to $\{e'_1, e'_2, \dots, e'_n\}$. Then, we can show that $x' = Tx$ (see Exercise 3.1).

Consider a linear transformation

$$\mathcal{L} : \mathbb{R}^n \rightarrow \mathbb{R}^n,$$

and let A be its representation with respect to $\{e_1, e_2, \dots, e_n\}$ and B its representation with respect to $\{e'_1, e'_2, \dots, e'_n\}$. Let $y = Ax$ and $y' = Bx$. Therefore, $y' = Ty = TAx = Bx = BTx$ and hence $TA = BT$, or $A = T^{-1}BT$.

Two $n \times n$ matrices A and B are *similar* if there exists a nonsingular matrix T such that $A = T^{-1}BT$. In conclusion, similar matrices correspond to the same linear transformation with respect to different bases.

3.2 Eigenvalues and Eigenvectors

Let A be an $n \times n$ real square matrix. A scalar λ (possibly complex) and a nonzero vector v satisfying the equation $Av = \lambda v$ are said to be, respectively, an *eigenvalue* and an *eigenvector* of A . For λ to be an eigenvalue it is necessary and sufficient for the matrix $\lambda I - A$ to be singular; that is, $\det[\lambda I - A] = 0$, where I is the $n \times n$ identity matrix. This leads to an n th-order polynomial equation

$$\det[\lambda I - A] = \lambda^n + a_{n-1}\lambda^{n-1} + \dots + a_1\lambda + a_0 = 0.$$

$$A^{\lambda} = \lambda v$$

We call the polynomial $\det[\lambda I - A]$ the *characteristic polynomial* of the matrix A , and the equation above the *characteristic equation*. According to the fundamental theorem of algebra, the characteristic equation must have n (possibly nondistinct) roots that are the eigenvalues of A . The following theorem states that if A has n distinct eigenvalues, then it also has n linearly independent eigenvectors.

$$\det(A - \lambda I) = 0$$

$$\lambda(v \neq 0) = \lambda v$$

Theorem 3.1 Suppose that the characteristic equation $\det[\lambda I - A] = 0$ has n distinct roots $\lambda_1, \lambda_2, \dots, \lambda_n$. Then, there exist n linearly independent vectors v_1, v_2, \dots, v_n such that

$$Av_i = \lambda_i v_i \quad i = 1, 2, \dots, n.$$

$$\lambda_1, \lambda_2, \dots, \lambda_n, v_1, v_2, \dots, v_n$$

Proof. Because $\det[\lambda_i \mathbf{I} - \mathbf{A}] = 0$, $i = 1, \dots, n$, there exist nonzero \mathbf{v}_i , $i = 1, \dots, n$, such that $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, $i = 1, \dots, n$. We now prove the linear independence of $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$. To do this, let c_1, \dots, c_n be scalars such that $\sum_{i=1}^n c_i \mathbf{v}_i = \mathbf{0}$. We show that $c_i = 0$, $i = 1, \dots, n$.

Consider the matrix

$$\mathbf{Z} = (\lambda_2 \mathbf{I} - \mathbf{A})(\lambda_3 \mathbf{I} - \mathbf{A}) \cdots (\lambda_n \mathbf{I} - \mathbf{A}).$$

We first show that $c_1 = 0$. Note that

$$\begin{aligned} \mathbf{Z}\mathbf{v}_n &= (\lambda_2 \mathbf{I} - \mathbf{A})(\lambda_3 \mathbf{I} - \mathbf{A}) \cdots (\lambda_{n-1} \mathbf{I} - \mathbf{A})(\lambda_n \mathbf{I} - \mathbf{A})\mathbf{v}_n \\ &= (\lambda_2 \mathbf{I} - \mathbf{A})(\lambda_3 \mathbf{I} - \mathbf{A}) \cdots (\lambda_{n-1} \mathbf{I} - \mathbf{A})(\lambda_n \mathbf{v}_n - \mathbf{A}\mathbf{v}_n) \\ &= \mathbf{0} \end{aligned}$$

since $\lambda_n \mathbf{v}_n - \mathbf{A}\mathbf{v}_n = \mathbf{0}$.

Repeating the argument above, we get

$$\mathbf{Z}\mathbf{v}_k = \mathbf{0}, \quad k = 2, 3, \dots, n.$$

But

$$\begin{aligned} \mathbf{Z}\mathbf{v}_1 &= (\lambda_2 \mathbf{I} - \mathbf{A})(\lambda_3 \mathbf{I} - \mathbf{A}) \cdots (\lambda_{n-1} \mathbf{I} - \mathbf{A})(\lambda_n \mathbf{I} - \mathbf{A})\mathbf{v}_1 \\ &= (\lambda_2 \mathbf{I} - \mathbf{A})(\lambda_3 \mathbf{I} - \mathbf{A}) \cdots (\lambda_{n-1} \mathbf{v}_1 - \mathbf{A}\mathbf{v}_1)(\lambda_n - \lambda_1) \\ &\quad \vdots \\ &= (\lambda_2 \mathbf{I} - \mathbf{A})(\lambda_3 \mathbf{I} - \mathbf{A})\mathbf{v}_1 \cdots (\lambda_{n-1} - \lambda_1)(\lambda_n - \lambda_1) \\ &= (\lambda_2 - \lambda_1)(\lambda_3 - \lambda_1) \cdots (\lambda_{n-1} - \lambda_1)(\lambda_n - \lambda_1)\mathbf{v}_1. \end{aligned}$$

Using the equation above, we see that

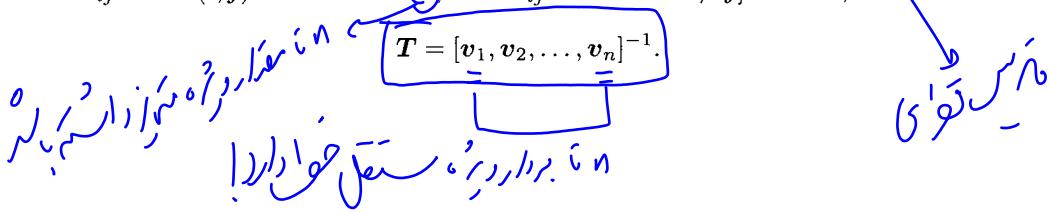
$$\begin{aligned} \mathbf{Z} \left(\sum_{i=1}^n c_i \mathbf{v}_i \right) &= \sum_{i=1}^n c_i \mathbf{Z}\mathbf{v}_i \\ &= c_1 \mathbf{Z}\mathbf{v}_1 \\ &= c_1 (\lambda_2 - \lambda_1)(\lambda_3 - \lambda_1) \cdots (\lambda_n - \lambda_1)\mathbf{v}_1 = \mathbf{0}. \end{aligned}$$

Because the λ_i are distinct, it must follow that $c_1 = 0$.

Using similar arguments, we can show that all c_i must vanish, and therefore the set of eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$ is linearly independent. ■

Consider a basis formed by a linearly independent set of eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$. With respect to this basis, the matrix $\boxed{\mathbf{A}}$ is *diagonal* [i.e., if a_{ij} is the (i, j) th element of $\boxed{\mathbf{A}}$ then $a_{ij} = 0$ for all $i \neq j$]. Indeed, let

$$\mathbf{T} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n]^{-1}.$$



Then,

$$\begin{aligned}
 TAT^{-1} &= TA[v_1, v_2, \dots, v_n] \\
 &= T[\underline{Av}_1, \underline{Av}_2, \dots, \underline{Av}_n] \\
 &= T[\lambda_1 v_1, \lambda_2 v_2, \dots, \lambda_n v_n] \\
 &= \boxed{TT^{-1}} \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix} \\
 &= \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix},
 \end{aligned}$$

because $TT^{-1} = I$.

A matrix \mathbf{A} is *symmetric* if $\mathbf{A} = \mathbf{A}^\top$.

Theorem 3.2 All eigenvalues of a real symmetric matrix are real.

Proof. Let

$$\mathbf{Ax} = \lambda\mathbf{x},$$

where $\mathbf{x} \neq \mathbf{0}$. Taking the inner product of \mathbf{Ax} with \mathbf{x} yields

$$\langle \mathbf{Ax}, \mathbf{x} \rangle = \langle \lambda\mathbf{x}, \mathbf{x} \rangle = \lambda\langle \mathbf{x}, \mathbf{x} \rangle.$$

On the other hand,

$$\langle \mathbf{Ax}, \mathbf{x} \rangle = \langle \mathbf{x}, \mathbf{A}^\top \mathbf{x} \rangle = \langle \mathbf{x}, \mathbf{Ax} \rangle = \langle \mathbf{x}, \lambda\mathbf{x} \rangle = \bar{\lambda}\langle \mathbf{x}, \mathbf{x} \rangle.$$

The above follows from the definition of the inner product on \mathbb{C}^n . We note that $\langle \mathbf{x}, \mathbf{x} \rangle$ is real and $\langle \mathbf{x}, \mathbf{x} \rangle > 0$. Hence,

$$\lambda\langle \mathbf{x}, \mathbf{x} \rangle = \bar{\lambda}\langle \mathbf{x}, \mathbf{x} \rangle$$

and

$$(\lambda - \bar{\lambda})\langle \mathbf{x}, \mathbf{x} \rangle = 0.$$

Because $\langle \mathbf{x}, \mathbf{x} \rangle > 0$,

$$\lambda = \bar{\lambda}.$$

Thus, λ is real.

Theorem 3.3 Any real symmetric $n \times n$ matrix has a set of n eigenvectors that are mutually orthogonal.

Proof. We prove the result for the case when the n eigenvalues are distinct. For a general proof, see [62, p. 104].

Suppose that $\mathbf{A}\mathbf{v}_1 = \lambda_1\mathbf{v}_1$, $\mathbf{A}\mathbf{v}_2 = \lambda_2\mathbf{v}_2$, where $\lambda_1 \neq \lambda_2$. Then,

$$\langle \mathbf{A}\mathbf{v}_1, \mathbf{v}_2 \rangle = \langle \lambda_1\mathbf{v}_1, \mathbf{v}_2 \rangle = \lambda_1 \langle \mathbf{v}_1, \mathbf{v}_2 \rangle.$$

Because $\mathbf{A} = \mathbf{A}^\top$,

$$\langle \mathbf{A}\mathbf{v}_1, \mathbf{v}_2 \rangle = \langle \mathbf{v}_1, \mathbf{A}^\top \mathbf{v}_2 \rangle = \langle \mathbf{v}_1, \mathbf{A}\mathbf{v}_2 \rangle = \lambda_2 \langle \mathbf{v}_1, \mathbf{v}_2 \rangle.$$

Therefore,

$$\lambda_1 \langle \mathbf{v}_1, \mathbf{v}_2 \rangle = \lambda_2 \langle \mathbf{v}_1, \mathbf{v}_2 \rangle.$$

Because $\lambda_1 \neq \lambda_2$, it follows that

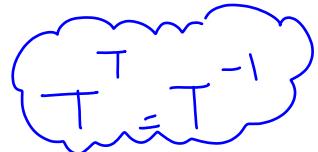
$$\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0.$$

If \mathbf{A} is symmetric, then a set of its eigenvectors forms an orthogonal basis for \mathbb{R}^n . If the basis $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$ is normalized so that each element has norm of unity, then defining the matrix

$$\mathbf{T} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n],$$

we have

$$\begin{array}{c} \text{new text} \\ \boxed{\mathbf{T}^\top \mathbf{T} = \mathbf{I}} \\ \text{and hence} \\ \boxed{\mathbf{T}^\top = \mathbf{T}^{-1}} \end{array}$$



A matrix whose transpose is its inverse is said to be an *orthogonal matrix*.

3.3 Orthogonal Projections

Recall that a subspace \mathcal{V} of \mathbb{R}^n is a subset that is closed under the operations of vector addition and scalar multiplication. In other words, \mathcal{V} is a subspace of \mathbb{R}^n if $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{V} \Rightarrow \alpha\mathbf{x}_1 + \beta\mathbf{x}_2 \in \mathcal{V}$ for all $\alpha, \beta \in \mathbb{R}$. Furthermore, the dimension of a subspace \mathcal{V} is equal to the maximum number of linearly independent vectors in \mathcal{V} . If \mathcal{V} is a subspace of \mathbb{R}^n , then the *orthogonal complement* of \mathcal{V} , denoted \mathcal{V}^\perp , consists of all vectors that are orthogonal to every vector in \mathcal{V} . Thus,

$$\mathcal{V}^\perp = \{\mathbf{x} : \mathbf{v}^\top \mathbf{x} = 0 \text{ for all } \mathbf{v} \in \mathcal{V}\}.$$

The orthogonal complement of \mathcal{V} is also a subspace (see Exercise 3.7). Together, \mathcal{V} and \mathcal{V}^\perp span \mathbb{R}^n in the sense that every vector $\mathbf{x} \in \mathbb{R}^n$ can be represented uniquely as

$$\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2,$$

new text $\mathbf{x}_1 \in \mathcal{V}, \mathbf{x}_2 \in \mathcal{V}^\perp$

where $\mathbf{x}_1 \in \mathcal{V}$ and $\mathbf{x}_2 \in \mathcal{V}^\perp$. We call the representation above the *orthogonal decomposition* of \mathbf{x} (with respect to \mathcal{V}). We say that \mathbf{x}_1 and \mathbf{x}_2 are *orthogonal projections* of \mathbf{x} onto the subspaces \mathcal{V} and \mathcal{V}^\perp , respectively. We write $\mathbb{R}^n = \mathcal{V} \oplus \mathcal{V}^\perp$ and say that \mathbb{R}^n is a *direct sum* of \mathcal{V} and \mathcal{V}^\perp . We say that a linear transformation \mathbf{P} is an *orthogonal projector* onto \mathcal{V} if for all $\mathbf{x} \in \mathbb{R}^n$, we have $\mathbf{Px} \in \mathcal{V}$ and $\mathbf{x} - \mathbf{Px} \in \mathcal{V}^\perp$.

In the subsequent discussion we use the following notation. Let $\mathbf{A} \in \mathbb{R}^{m \times n}$. Let the *range*, or *image*, of \mathbf{A} be denoted

$$\mathcal{R}(\mathbf{A}) \triangleq \{\mathbf{Ax} : \mathbf{x} \in \mathbb{R}^n\},$$

and the *nullspace*, or *kernel*, of \mathbf{A} be denoted

$$\mathcal{N}(\mathbf{A}) \triangleq \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{0}\}.$$

Note that $\mathcal{R}(\mathbf{A})$ and $\mathcal{N}(\mathbf{A})$ are subspaces (see Exercise 3.9)

Theorem 3.4 Let \mathbf{A} be a given matrix. Then, $\mathcal{R}(\mathbf{A})^\perp = \mathcal{N}(\mathbf{A}^\top)$ and $\mathcal{N}(\mathbf{A})^\perp = \mathcal{R}(\mathbf{A}^\top)$. \square

Proof. Suppose that $\mathbf{x} \in \mathcal{R}(\mathbf{A})^\perp$. Then, $\mathbf{y}^\top (\mathbf{A}^\top \mathbf{x}) = (\mathbf{Ay})^\top \mathbf{x} = 0$ for all \mathbf{y} , so that $\mathbf{A}^\top \mathbf{x} = \mathbf{0}$. Hence, $\mathbf{x} \in \mathcal{N}(\mathbf{A}^\top)$. This implies that $\mathcal{R}(\mathbf{A})^\perp \subset \mathcal{N}(\mathbf{A}^\top)$.

If now $\mathbf{x} \in \mathcal{N}(\mathbf{A}^\top)$, then $(\mathbf{Ay})^\top \mathbf{x} = \mathbf{y}^\top (\mathbf{A}^\top \mathbf{x}) = 0$ for all \mathbf{y} , so that $\mathbf{x} \in \mathcal{R}(\mathbf{A})^\perp$, and consequently, $\mathcal{N}(\mathbf{A}^\top) \subset \mathcal{R}(\mathbf{A})^\perp$. Thus, $\mathcal{R}(\mathbf{A})^\perp = \mathcal{N}(\mathbf{A}^\top)$.

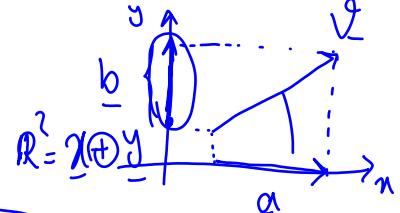
The equation $\mathcal{N}(\mathbf{A})^\perp = \mathcal{R}(\mathbf{A}^\top)$ follows from what we have proved above and the fact that for any subspace \mathcal{V} , we have $(\mathcal{V}^\perp)^\perp = \mathcal{V}$ (see Exercise 3.11). \blacksquare

Theorem 3.4 allows us to establish the following necessary and sufficient condition for orthogonal projectors. For this, note that if \mathbf{P} is an orthogonal projector onto \mathcal{V} , then $\mathbf{Px} = \mathbf{x}$ for all $\mathbf{x} \in \mathcal{V}$, and $\mathcal{R}(\mathbf{P}) = \mathcal{V}$ (see Exercise 3.14).

Theorem 3.5 A matrix \mathbf{P} is an orthogonal projector [onto the subspace $\mathcal{V} = \mathcal{R}(\mathbf{P})$] if and only if $\mathbf{P}^2 = \mathbf{P} = \mathbf{P}^\top$. \square

Proof. \Rightarrow : Suppose that \mathbf{P} is an orthogonal projector onto $\mathcal{V} = \mathcal{R}(\mathbf{P})$. Then, $\mathcal{R}(\mathbf{I} - \mathbf{P}) \subset \mathcal{R}(\mathbf{P})^\perp$. But, $\mathcal{R}(\mathbf{P})^\perp = \mathcal{N}(\mathbf{P}^\top)$ by Theorem 3.4. Therefore, $\mathcal{R}(\mathbf{I} - \mathbf{P}) \subset \mathcal{N}(\mathbf{P}^\top)$. Hence, $\mathbf{P}^\top (\mathbf{I} - \mathbf{P}) \mathbf{y} = \mathbf{0}$ for all \mathbf{y} , which implies that $\mathbf{P}^\top (\mathbf{I} - \mathbf{P}) = \mathbf{O}$, where \mathbf{O} is the matrix with all entries equal to zero; i.e., the zero matrix. Therefore, $\mathbf{P}^\top = \mathbf{P}^\top \mathbf{P}$, and thus $\mathbf{P} = \mathbf{P}^\top = \mathbf{P}^2$.

\Leftarrow : Suppose that $\mathbf{P}^2 = \mathbf{P} = \mathbf{P}^\top$. For any \mathbf{x} , we have $(\mathbf{Py})^\top (\mathbf{I} - \mathbf{P}) \mathbf{x} = \mathbf{y}^\top \mathbf{P}^\top (\mathbf{I} - \mathbf{P}) \mathbf{x} = \mathbf{y}^\top \mathbf{P} (\mathbf{I} - \mathbf{P}) \mathbf{x} = 0$ for all \mathbf{y} . Thus, $(\mathbf{I} - \mathbf{P}) \mathbf{x} \in \mathcal{R}(\mathbf{P})^\perp$, which means that \mathbf{P} is an orthogonal projector. \blacksquare



3.4 Quadratic Forms

A *quadratic form* $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a function

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{Q} \mathbf{x},$$

where \mathbf{Q} is an $n \times n$ real matrix. There is no loss of generality in assuming \mathbf{Q} to be symmetric: $\mathbf{Q} = \mathbf{Q}^\top$. For if the matrix \mathbf{Q} is not symmetric, we can always replace it with the symmetric matrix

$$\mathbf{Q}_0 = \mathbf{Q}_0^\top = \frac{1}{2} (\mathbf{Q} + \mathbf{Q}^\top).$$

Note that

$$\mathbf{x}^\top \mathbf{Q} \mathbf{x} = \mathbf{x}^\top \mathbf{Q}_0 \mathbf{x} = \mathbf{x}^\top \left(\frac{1}{2} \mathbf{Q} + \frac{1}{2} \mathbf{Q}^\top \right) \mathbf{x}.$$

A quadratic form $\mathbf{x}^\top \mathbf{Q} \mathbf{x}$, $\mathbf{Q} = \mathbf{Q}^\top$, is said to be *positive definite* if $\mathbf{x}^\top \mathbf{Q} \mathbf{x} > 0$ for all nonzero vectors \mathbf{x} . It is *positive semidefinite* if $\mathbf{x}^\top \mathbf{Q} \mathbf{x} \geq 0$ for all \mathbf{x} . Similarly, we define the quadratic form to be *negative definite*, or *negative semidefinite*, if $\mathbf{x}^\top \mathbf{Q} \mathbf{x} < 0$ for all nonzero vectors \mathbf{x} , or $\mathbf{x}^\top \mathbf{Q} \mathbf{x} \leq 0$ for all \mathbf{x} , respectively.

Recall that the minors of a matrix \mathbf{Q} are the determinants of the matrices obtained by successively removing rows and columns from \mathbf{Q} . The *principal minors* are $\det \mathbf{Q}$ itself and the determinants of matrices obtained by successively removing an i th row and an i th column. That is, the principal minors are

$$\det \begin{bmatrix} q_{i_1 i_1} & q_{i_1 i_2} & \cdots & q_{i_1 i_p} \\ q_{i_2 i_1} & q_{i_2 i_2} & \cdots & q_{i_2 i_p} \\ \vdots & \vdots & & \vdots \\ q_{i_p i_1} & q_{i_p i_2} & \cdots & q_{i_p i_p} \end{bmatrix}, \quad 1 \leq i_1 < \cdots < i_p \leq n, \quad p = 1, 2, \dots, n.$$

The *leading principal minors* are $\det \mathbf{Q}$ and the minors obtained by successively removing the last row and the last column. That is, the leading principal minors are

$$\begin{aligned} \Delta_1 &= q_{11}, & \Delta_2 &= \det \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix}, \\ \Delta_3 &= \det \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}, & \dots, & \Delta_n = \det \mathbf{Q}. \end{aligned}$$

We now prove *Sylvester's criterion*, which allows us to determine if a quadratic form $\mathbf{x}^\top \mathbf{Q} \mathbf{x}$ is positive definite using only the leading principal minors of \mathbf{Q} .

Theorem 3.6 Sylvester's Criterion. A quadratic form $\mathbf{x}^\top \mathbf{Q}\mathbf{x}$, $\mathbf{Q} = \mathbf{Q}^\top$, is positive definite if and only if the leading principal minors of \mathbf{Q} are positive.

□

Proof. The key to the proof of Sylvester's criterion is the fact that a quadratic form whose leading principal minors are nonzero can be expressed in some basis as a sum of squares

$$\frac{\Delta_0}{\Delta_1} \tilde{x}_1^2 + \frac{\Delta_1}{\Delta_2} \tilde{x}_2^2 + \cdots + \frac{\Delta_{n-1}}{\Delta_n} \tilde{x}_n^2,$$

where \tilde{x}_i are the coordinates of the vector \mathbf{x} in the new basis, $\Delta_0 \triangleq 1$, and $\Delta_1, \dots, \Delta_n$ are the leading principal minors of \mathbf{Q} .

To this end, consider a quadratic form $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{Q}\mathbf{x}$, where $\mathbf{Q} = \mathbf{Q}^\top$. Let $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n\}$ be the natural basis for \mathbb{R}^n , and let

$$\mathbf{x} = x_1 \mathbf{e}_1 + x_2 \mathbf{e}_2 + \cdots + x_n \mathbf{e}_n$$

be a given vector in \mathbb{R}^n . Let $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$ be another basis for \mathbb{R}^n . Then, the vector \mathbf{x} is represented in the new basis as $\tilde{\mathbf{x}}$, where

$$\mathbf{x} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n] \tilde{\mathbf{x}} \triangleq \mathbf{V} \tilde{\mathbf{x}}.$$

Accordingly, the quadratic form can be written as

$$\mathbf{x}^\top \mathbf{Q}\mathbf{x} = \tilde{\mathbf{x}}^\top \mathbf{V}^\top \mathbf{Q} \mathbf{V} \tilde{\mathbf{x}} = \tilde{\mathbf{x}}^\top \tilde{\mathbf{Q}} \tilde{\mathbf{x}},$$

where

$$\tilde{\mathbf{Q}} = \mathbf{V}^\top \mathbf{Q} \mathbf{V} = \begin{bmatrix} \tilde{q}_{11} & \cdots & \tilde{q}_{1n} \\ \vdots & \ddots & \vdots \\ \tilde{q}_{n1} & \cdots & \tilde{q}_{nn} \end{bmatrix}.$$

Note that $\tilde{q}_{ij} = \langle \mathbf{v}_i, \mathbf{Q}\mathbf{v}_j \rangle$. Our goal is to determine conditions on the new basis $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$ such that $\tilde{q}_{ij} = 0$ for $i \neq j$.

We seek the new basis in the form

$$\begin{aligned} \mathbf{v}_1 &= \alpha_{11} \mathbf{e}_1, \\ \mathbf{v}_2 &= \alpha_{21} \mathbf{e}_1 + \alpha_{22} \mathbf{e}_2, \\ &\vdots \\ \mathbf{v}_n &= \alpha_{n1} \mathbf{e}_1 + \alpha_{n2} \mathbf{e}_2 + \cdots + \alpha_{nn} \mathbf{e}_n. \end{aligned}$$

Observe that for $j = 1, \dots, i-1$, if

$$\langle \mathbf{v}_i, \mathbf{Q}\mathbf{e}_j \rangle = 0,$$

then

$$\langle \mathbf{v}_i, \mathbf{Q}\mathbf{v}_j \rangle = 0.$$

Our goal then is to determine the coefficients $\alpha_{i1}, \alpha_{i2}, \dots, \alpha_{ii}$, $i = 1, \dots, n$, such that the vector

$$\mathbf{v}_i = \alpha_{i1}\mathbf{e}_1 + \alpha_{i2}\mathbf{e}_2 + \cdots + \alpha_{ii}\mathbf{e}_i$$

satisfies the i relations

$$\begin{aligned} \langle \mathbf{v}_i, \mathbf{Q}\mathbf{e}_j \rangle &= 0, \quad j = 1, \dots, i-1, \\ \langle \mathbf{e}_i, \mathbf{Q}\mathbf{v}_i \rangle &= 1. \end{aligned}$$

In this case, we get

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \alpha_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \alpha_{nn} \end{bmatrix}.$$

For each $i = 1, \dots, n$, the i relations above determine the coefficients $\alpha_{i1}, \dots, \alpha_{ii}$ in a unique way. Indeed, upon substituting the expression for \mathbf{v}_i into the equations above, we obtain the set of equations

$$\alpha_{i1}q_{11} + \alpha_{i2}q_{12} + \cdots + \alpha_{ii}q_{1i} = 0,$$

$$\vdots$$

$$\alpha_{i1}q_{i-11} + \alpha_{i2}q_{i-12} + \cdots + \alpha_{ii}q_{i-1i} = 0,$$

$$\alpha_{i1}q_{ii} + \alpha_{i2}q_{i2} + \cdots + \alpha_{ii}q_{ii} = 1.$$

The set of equations above can be expressed in matrix form as

$$\begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1i} \\ q_{21} & q_{22} & \cdots & q_{2i} \\ \vdots & \vdots & \ddots & \vdots \\ q_{i1} & q_{i2} & \cdots & q_{ii} \end{bmatrix} \begin{bmatrix} \alpha_{i1} \\ \alpha_{i2} \\ \vdots \\ \alpha_{ii} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}.$$

If the leading principal minors of the matrix \mathbf{Q} do not vanish, then the coefficients α_{ij} can be obtained using *Cramer's rule*. In particular,

$$\alpha_{ii} = \frac{1}{\Delta_i} \det \begin{bmatrix} q_{11} & \cdots & q_{1i-1} & 0 \\ \vdots & \ddots & \vdots & 0 \\ q_{i-11} & \cdots & q_{i-1i-1} & 0 \\ q_{ii} & \cdots & q_{ii-1} & 1 \end{bmatrix} = \frac{\Delta_{i-1}}{\Delta_i}.$$

Hence,

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \frac{1}{\Delta_1} & & & 0 \\ & \frac{\Delta_1}{\Delta_2} & & \\ & & \ddots & \\ 0 & & & \frac{\Delta_{n-1}}{\Delta_n} \end{bmatrix}.$$

In the new basis, the quadratic form can be expressed as a sum of squares

$$\mathbf{x}^\top \mathbf{Q} \mathbf{x} = \tilde{\mathbf{x}}^\top \tilde{\mathbf{Q}} \tilde{\mathbf{x}} = \frac{1}{\Delta_1} \tilde{x}_1^2 + \frac{\Delta_1}{\Delta_2} \tilde{x}_2^2 + \cdots + \frac{\Delta_{n-1}}{\Delta_n} \tilde{x}_n^2.$$

We now show that a necessary and sufficient condition for the quadratic form to be positive definite is $\Delta_i > 0$, $i = 1, \dots, n$.

Sufficiency is clear, for if $\Delta_i > 0$, $i = 1, \dots, n$, then by the previous argument there is a basis such that

$$\mathbf{x}^\top \mathbf{Q} \mathbf{x} = \tilde{\mathbf{x}}^\top \tilde{\mathbf{Q}} \tilde{\mathbf{x}} > 0$$

for any $\mathbf{x} \neq \mathbf{0}$ (or, equivalently, any $\tilde{\mathbf{x}} \neq \mathbf{0}$).

To prove necessity, we first show that for $i = 1, \dots, n$, we have $\Delta_i \neq 0$. To see this, suppose that $\Delta_k = 0$ for some k . Note that $\Delta_k = \det \mathbf{Q}_k$,

$$\mathbf{Q}_k = \begin{bmatrix} q_{11} & \cdots & q_{1k} \\ \vdots & \ddots & \vdots \\ q_{k1} & \cdots & q_{kk} \end{bmatrix}.$$

Then, there exists a vector $\mathbf{v} \in \mathbb{R}^k$, $\mathbf{v} \neq \mathbf{0}$, such that $\mathbf{v}^\top \mathbf{Q}_k = \mathbf{0}$. Now let $\mathbf{x} \in \mathbb{R}^n$ be given by $\mathbf{x} = [\mathbf{v}^\top, \mathbf{0}^\top]^\top$. Then,

$$\mathbf{x}^\top \mathbf{Q} \mathbf{x} = \mathbf{v}^\top \mathbf{Q}_k \mathbf{v} = 0.$$

But $\mathbf{x} \neq \mathbf{0}$, which contradicts the fact that the quadratic form f is positive definite. Therefore, if $\mathbf{x}^\top \mathbf{Q} \mathbf{x} > 0$, then $\Delta_i \neq 0$, $i = 1, \dots, n$. Then, using our previous argument, we may write

$$\mathbf{x}^\top \mathbf{Q} \mathbf{x} = \tilde{\mathbf{x}}^\top \tilde{\mathbf{Q}} \tilde{\mathbf{x}} = \frac{1}{\Delta_1} \tilde{x}_1^2 + \frac{\Delta_1}{\Delta_2} \tilde{x}_2^2 + \cdots + \frac{\Delta_{n-1}}{\Delta_n} \tilde{x}_n^2,$$

where $\tilde{\mathbf{x}} = [\mathbf{v}_1, \dots, \mathbf{v}_n] \mathbf{x}$. Hence, if the quadratic form is positive definite, then all leading principal minors must be positive. ■

Note that if \mathbf{Q} is not symmetric, Sylvester's criterion cannot be used to check positive definiteness of the quadratic form $\mathbf{x}^\top \mathbf{Q} \mathbf{x}$. To see this, consider an example where

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 \\ -4 & 1 \end{bmatrix}.$$

The leading principal minors of \mathbf{Q} are $\Delta_1 = 1 > 0$ and $\Delta_2 = \det \mathbf{Q} = 1 > 0$. However, if $\mathbf{x} = [1, 1]^\top$, then $\mathbf{x}^\top \mathbf{Q} \mathbf{x} = -2 < 0$, and hence the associated quadratic form is not positive definite. Note that

$$\begin{aligned} \mathbf{x}^\top \mathbf{Q} \mathbf{x} &= \mathbf{x}^\top \begin{bmatrix} 1 & 0 \\ -4 & 1 \end{bmatrix} \mathbf{x} = \frac{1}{2} \mathbf{x}^\top \left(\begin{bmatrix} 1 & 0 \\ -4 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -4 \\ 0 & 1 \end{bmatrix} \right) \mathbf{x} \\ &= \mathbf{x}^\top \begin{bmatrix} 1 & -2 \\ -2 & 1 \end{bmatrix} \mathbf{x} = \mathbf{x}^\top \mathbf{Q}_0 \mathbf{x}. \end{aligned}$$

The leading principal minors of \mathbf{Q}_0 are $\Delta_1 = 1 > 0$ and $\Delta_2 = \det \mathbf{Q}_0 = -3 < 0$, as expected.

A necessary condition for a real quadratic form to be positive semidefinite is that the leading principal minors be nonnegative. However, this is *not* a sufficient condition (see Exercise 3.16). In fact, a real quadratic form is positive semidefinite if and only if all principal minors are nonnegative (for a proof of this fact, see [44, p. 307]).

A symmetric matrix \mathbf{Q} is said to be *positive definite* if the quadratic form $\mathbf{x}^\top \mathbf{Q} \mathbf{x}$ is positive definite. If \mathbf{Q} is positive definite, we write $\mathbf{Q} > 0$. Similarly, we define a symmetric matrix \mathbf{Q} to be *positive semidefinite* ($\mathbf{Q} \geq 0$), *negative definite* ($\mathbf{Q} < 0$), and *negative semidefinite* ($\mathbf{Q} \leq 0$) if the corresponding quadratic forms have the respective properties. The symmetric matrix \mathbf{Q} is *indefinite* if it is neither positive semidefinite nor negative semidefinite. Note that the matrix \mathbf{Q} is positive definite (semidefinite) if and only if the matrix $-\mathbf{Q}$ is negative definite (semidefinite).

Sylvester's criterion provides a way of checking the definiteness of a quadratic form, or equivalently, a symmetric matrix. An alternative method involves checking the eigenvalues of \mathbf{Q} , as stated below.

Theorem 3.7 *A symmetric matrix \mathbf{Q} is positive definite (or positive semidefinite) if and only if all eigenvalues of \mathbf{Q} are positive (or nonnegative).* \square

Proof. For any \mathbf{x} , let $\mathbf{y} = \mathbf{T}^{-1}\mathbf{x} = \mathbf{T}^\top \mathbf{x}$, where \mathbf{T} is an orthogonal matrix whose columns are eigenvectors of \mathbf{Q} . Then, $\mathbf{x}^\top \mathbf{Q} \mathbf{x} = \mathbf{y}^\top \mathbf{T}^\top \mathbf{Q} \mathbf{T} \mathbf{y} = \sum_{i=1}^n \lambda_i y_i^2$. From this, the result follows. \blacksquare

Through diagonalization, we can show that a symmetric positive semidefinite matrix \mathbf{Q} has a positive semidefinite (symmetric) square root $\mathbf{Q}^{1/2}$ satisfying $\mathbf{Q}^{1/2} \mathbf{Q}^{1/2} = \mathbf{Q}$. For this, we use \mathbf{T} as above and define

$$\mathbf{Q}^{1/2} = \mathbf{T} \begin{bmatrix} \lambda_1^{1/2} & & & 0 \\ & \lambda_2^{1/2} & & \\ & & \ddots & \\ 0 & & & \lambda_n^{1/2} \end{bmatrix} \mathbf{T}^\top,$$

which is easily verified to have the desired properties. Note that the quadratic form $\mathbf{x}^\top \mathbf{Q} \mathbf{x}$ can be expressed as $\|\mathbf{Q}^{1/2} \mathbf{x}\|^2$.

In summary, we have presented two tests for definiteness of quadratic forms and symmetric matrices. We point out again that nonnegativity of leading principal minors is a necessary but not a sufficient condition for positive semidefiniteness.

3.5 Matrix Norms

The norm of a matrix may be chosen in a variety of ways. Because the set of matrices $\mathbb{R}^{m \times n}$ can be viewed as the real vector space \mathbb{R}^{mn} , matrix norms

should be no different from regular vector norms. Therefore, we define the norm of a matrix \mathbf{A} , denoted $\|\mathbf{A}\|$, to be any function $\|\cdot\|$ that satisfies the following conditions:

1. $\|\mathbf{A}\| > 0$ if $\mathbf{A} \neq \mathbf{O}$, and $\|\mathbf{O}\| = 0$, where \mathbf{O} is a matrix with all entries equal to zero.
2. $\|c\mathbf{A}\| = |c|\|\mathbf{A}\|$, for any $c \in \mathbb{R}$.
3. $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$.

An example of a matrix norm is the *Frobenius norm*, defined as

$$\|\mathbf{A}\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n (a_{ij})^2 \right)^{\frac{1}{2}},$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$. Note that the Frobenius norm is equivalent to the Euclidean norm on \mathbb{R}^{mn} .

For our purposes, we consider only matrix norms that satisfy the following additional condition:

4. $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \|\mathbf{B}\|$.

It turns out that the Frobenius norm satisfies condition 4 as well.

In many problems, both matrices and vectors appear simultaneously. Therefore, it is convenient to construct the norm of a matrix in such a way that it will be related to vector norms. To this end we consider a special class of matrix norms, called *induced norms*. Let $\|\cdot\|_{(n)}$ and $\|\cdot\|_{(m)}$ be vector norms on \mathbb{R}^n and \mathbb{R}^m , respectively. We say that the matrix norm is *induced* by, or is *compatible* with, the given vector norms if for any matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and any vector $\mathbf{x} \in \mathbb{R}^n$, the following inequality is satisfied:

$$\|\mathbf{Ax}\|_{(m)} \leq \|\mathbf{A}\| \|\mathbf{x}\|_{(n)}.$$

We can define an induced matrix norm as

$$\|\mathbf{A}\| = \max_{\|\mathbf{x}\|_{(n)}=1} \|\mathbf{Ax}\|_{(m)};$$

that is, $\|\mathbf{A}\|$ is the maximum of the norms of the vectors \mathbf{Ax} where the vector \mathbf{x} runs over the set of all vectors with unit norm. When there is no ambiguity, we omit the subscripts (m) and (n) from $\|\cdot\|_{(m)}$ and $\|\cdot\|_{(n)}$.

Because of the continuity of a vector norm (see Exercise 2.10), for each matrix \mathbf{A} the maximum

$$\max_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\|$$

is attainable; that is, a vector \mathbf{x}_0 exists such that $\|\mathbf{x}_0\| = 1$ and $\|\mathbf{Ax}_0\| = \|\mathbf{A}\|$. This fact follows from the theorem of Weierstrass (see Theorem 4.2).

The induced norm satisfies conditions 1 to 4 and the compatibility condition, as we prove below.

Proof of Condition 1. Let $\mathbf{A} \neq \mathbf{O}$. Then, a vector \mathbf{x} , $\|\mathbf{x}\| = 1$, can be found such that $\mathbf{Ax} \neq \mathbf{0}$, and thus $\|\mathbf{Ax}\| \neq 0$. Hence, $\|\mathbf{A}\| = \max_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\| \neq 0$. If, on the other hand, $\mathbf{A} = \mathbf{O}$, then $\|\mathbf{A}\| = \max_{\|\mathbf{x}\|=1} \|\mathbf{Ox}\| = 0$. ■

Proof of Condition 2. By definition, $\|c\mathbf{A}\| = \max_{\|\mathbf{x}\|=1} \|c\mathbf{Ax}\|$. Obviously, $\|c\mathbf{Ax}\| = |c|\|\mathbf{Ax}\|$, and therefore $\|c\mathbf{A}\| = \max_{\|\mathbf{x}\|=1} |c|\|\mathbf{Ax}\| = |c| \max_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\| = |c|\|\mathbf{A}\|$. ■

Proof of Compatibility Condition. Let $\mathbf{y} \neq \mathbf{0}$ be any vector. Then, $\mathbf{x} = \mathbf{y}/\|\mathbf{y}\|$ satisfies the condition $\|\mathbf{x}\| = 1$. Consequently, $\|\mathbf{Ay}\| = \|\mathbf{A}(\|\mathbf{y}\|\mathbf{x})\| = \|\mathbf{y}\|\|\mathbf{Ax}\| \leq \|\mathbf{y}\|\|\mathbf{A}\|$. ■

Proof of Condition 3. For the matrix $\mathbf{A} + \mathbf{B}$, we can find a vector \mathbf{x}_0 such that $\|\mathbf{A} + \mathbf{B}\| = \|(\mathbf{A} + \mathbf{B})\mathbf{x}_0\|$ and $\|\mathbf{x}_0\| = 1$. Then, we have

$$\begin{aligned}\|\mathbf{A} + \mathbf{B}\| &= \|(\mathbf{A} + \mathbf{B})\mathbf{x}_0\| \\ &= \|\mathbf{Ax}_0 + \mathbf{Bx}_0\| \\ &\leq \|\mathbf{Ax}_0\| + \|\mathbf{Bx}_0\| \\ &\leq \|\mathbf{A}\|\|\mathbf{x}_0\| + \|\mathbf{B}\|\|\mathbf{x}_0\| \\ &= \|\mathbf{A}\| + \|\mathbf{B}\|,\end{aligned}$$

which shows that condition 3 holds. ■

Proof of Condition 4. For the matrix \mathbf{AB} , we can find a vector \mathbf{x}_0 such that $\|\mathbf{x}_0\| = 1$ and $\|\mathbf{ABx}_0\| = \|\mathbf{AB}\|$. Then, we have

$$\begin{aligned}\|\mathbf{AB}\| &= \|\mathbf{ABx}_0\| \\ &= \|\mathbf{A}(\mathbf{Bx}_0)\| \\ &\leq \|\mathbf{A}\|\|\mathbf{Bx}_0\| \\ &\leq \|\mathbf{A}\|\|\mathbf{B}\|\|\mathbf{x}_0\| \\ &= \|\mathbf{A}\|\|\mathbf{B}\|,\end{aligned}$$

which shows that condition 4 holds. ■

Theorem 3.8 *Let*

$$\|\mathbf{x}\| = \left(\sum_{k=1}^n |x_k|^2 \right)^{1/2} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}.$$

The matrix norm induced by this vector norm is

$$\|\mathbf{A}\| = \sqrt{\lambda_1},$$

where λ_1 is the largest eigenvalue of the matrix $\mathbf{A}^\top \mathbf{A}$. \square

Proof. We have

$$\|\mathbf{A}\mathbf{x}\|^2 = \langle \mathbf{A}\mathbf{x}, \mathbf{A}\mathbf{x} \rangle = \langle \mathbf{x}, \mathbf{A}^\top \mathbf{A}\mathbf{x} \rangle.$$

The matrix $\mathbf{A}^\top \mathbf{A}$ is symmetric and positive semidefinite. Let $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$ be its eigenvalues and $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ the orthonormal set of the eigenvectors corresponding to these eigenvalues. Now, we take an arbitrary vector \mathbf{x} with $\|\mathbf{x}\| = 1$ and represent it as a linear combination of \mathbf{x}_i , $i = 1, \dots, n$:

$$\mathbf{x} = c_1 \mathbf{x}_1 + c_2 \mathbf{x}_2 + \dots + c_n \mathbf{x}_n.$$

Note that

$$\langle \mathbf{x}, \mathbf{x} \rangle = c_1^2 + c_2^2 + \dots + c_n^2 = 1.$$

Furthermore,

$$\begin{aligned} \|\mathbf{A}\mathbf{x}\|^2 &= \langle \mathbf{x}, \mathbf{A}^\top \mathbf{A}\mathbf{x} \rangle \\ &= \langle c_1 \mathbf{x}_1 + \dots + c_n \mathbf{x}_n, c_1 \lambda_1 \mathbf{x}_1 + \dots + c_n \lambda_n \mathbf{x}_n \rangle \\ &= \lambda_1 c_1^2 + \dots + \lambda_n c_n^2 \\ &\leq \lambda_1 (c_1^2 + \dots + c_n^2) \\ &= \lambda_1. \end{aligned}$$

For the eigenvector \mathbf{x}_1 of $\mathbf{A}^\top \mathbf{A}$ corresponding to the eigenvalue λ_1 , we have

$$\|\mathbf{A}\mathbf{x}_1\|^2 = \langle \mathbf{x}_1, \mathbf{A}^\top \mathbf{A}\mathbf{x}_1 \rangle = \langle \mathbf{x}_1, \lambda_1 \mathbf{x}_1 \rangle = \lambda_1,$$

and hence

$$\max_{\|\mathbf{x}\|=1} \|\mathbf{A}\mathbf{x}\| = \sqrt{\lambda_1}.$$

This completes the proof. \blacksquare

Using arguments similar to the above, we can deduce the following important inequalities.

Rayleigh's Inequalities. If an $n \times n$ matrix \mathbf{P} is real symmetric positive definite, then

$$\lambda_{\min}(\mathbf{P})\|\mathbf{x}\|^2 \leq \mathbf{x}^\top \mathbf{P}\mathbf{x} \leq \lambda_{\max}(\mathbf{P})\|\mathbf{x}\|^2,$$

where $\lambda_{\min}(\mathbf{P})$ denotes the smallest eigenvalue of \mathbf{P} , and $\lambda_{\max}(\mathbf{P})$ denotes the largest eigenvalue of \mathbf{P} .

Example 3.1 Consider the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix},$$

and let the norm in \mathbb{R}^2 be given by

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2}.$$

Then,

$$\mathbf{A}^\top \mathbf{A} = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix},$$

and $\det[\lambda \mathbf{I}_2 - \mathbf{A}^\top \mathbf{A}] = \lambda^2 - 10\lambda + 9 = (\lambda - 1)(\lambda - 9)$. Thus, $\|\mathbf{A}\| = \sqrt{9} = 3$.

The eigenvector of $\mathbf{A}^\top \mathbf{A}$ corresponding to $\lambda_1 = 9$ is

$$\mathbf{x}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Note that $\|\mathbf{A}\mathbf{x}_1\| = \|\mathbf{A}\|$. Indeed,

$$\begin{aligned} \|\mathbf{A}\mathbf{x}_1\| &= \left\| \frac{1}{\sqrt{2}} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\| \\ &= \frac{1}{\sqrt{2}} \left\| \begin{bmatrix} 3 \\ 3 \end{bmatrix} \right\| \\ &= \frac{1}{\sqrt{2}} \sqrt{3^2 + 3^2} \\ &= 3. \end{aligned}$$

Because $\mathbf{A} = \mathbf{A}^\top$ in this example, we also have $\|\mathbf{A}\| = \max_{1 \leq i \leq n} |\lambda_i(\mathbf{A})|$, where $\lambda_1(\mathbf{A}), \dots, \lambda_n(\mathbf{A})$ are the eigenvalues of \mathbf{A} (possibly repeated). ■

Warning: In general, $\max_{1 \leq i \leq n} |\lambda_i(\mathbf{A})| \neq \|\mathbf{A}\|$. Instead, we have $\|\mathbf{A}\| \geq \max_{1 \leq i \leq n} |\lambda_i(\mathbf{A})|$, as illustrated in the following example (see also Exercise 5.2).

Example 3.2 Let

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix};$$

then

$$\mathbf{A}^\top \mathbf{A} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

and

$$\det[\lambda \mathbf{I}_2 - \mathbf{A}^\top \mathbf{A}] = \det \begin{bmatrix} \lambda & 0 \\ 0 & \lambda - 1 \end{bmatrix} = \lambda(\lambda - 1).$$

Note that 0 is the only eigenvalue of \mathbf{A} . Thus, for $i = 1, 2$, $\|\mathbf{A}\| = 1 > |\lambda_i(\mathbf{A})| = 0$. ■

For a more complete but still basic treatment of topics in linear algebra as discussed in this and the preceding chapter, see [47], [66], [95], [126]. For a treatment of matrices, we refer the reader to [44], [62]. Numerical aspects of matrix computations are discussed in [41], [53].

EXERCISES

3.1 Fix a vector in \mathbb{R}^n . Let \mathbf{x} be the column of the coordinates of the vector with respect to the basis $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n\}$ and \mathbf{x}' the coordinates of the same vector with respect to the basis $\{\mathbf{e}'_1, \mathbf{e}'_2, \dots, \mathbf{e}'_n\}$. Show that $\mathbf{x}' = \mathbf{T}\mathbf{x}$, where \mathbf{T} is the transformation matrix from $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n\}$ to $\{\mathbf{e}'_1, \mathbf{e}'_2, \dots, \mathbf{e}'_n\}$.

3.2 For each of the following cases, find the transformation matrix \mathbf{T} from $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ to $\{\mathbf{e}'_1, \mathbf{e}'_2, \mathbf{e}'_3\}$:

- a. $\mathbf{e}'_1 = \mathbf{e}_1 + 3\mathbf{e}_2 - 4\mathbf{e}_3$, $\mathbf{e}'_2 = 2\mathbf{e}_1 - \mathbf{e}_2 + 5\mathbf{e}_3$, $\mathbf{e}'_3 = 4\mathbf{e}_1 + 5\mathbf{e}_2 + 3\mathbf{e}_3$.
- b. $\mathbf{e}_1 = \mathbf{e}'_1 + \mathbf{e}'_2 + 3\mathbf{e}'_3$, $\mathbf{e}_2 = 2\mathbf{e}'_1 - \mathbf{e}'_2 + 4\mathbf{e}'_3$, $\mathbf{e}_3 = 3\mathbf{e}'_1 + 5\mathbf{e}'_3$.

3.3 Consider two bases of \mathbb{R}^3 , $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ and $\{\mathbf{e}'_1, \mathbf{e}'_2, \mathbf{e}'_3\}$, where $\mathbf{e}_1 = 2\mathbf{e}'_1 + \mathbf{e}'_2 - \mathbf{e}'_3$, $\mathbf{e}_2 = 2\mathbf{e}'_1 - \mathbf{e}'_2 + 2\mathbf{e}'_3$, and $\mathbf{e}_3 = 3\mathbf{e}'_1 + \mathbf{e}'_3$. Suppose that a linear transformation has a matrix representation in $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ of the form

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Find the matrix representation of this linear transformation in the basis $\{\mathbf{e}'_1, \mathbf{e}'_2, \mathbf{e}'_3\}$.

3.4 Consider two bases of \mathbb{R}^4 , $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4\}$ and $\{\mathbf{e}'_1, \mathbf{e}'_2, \mathbf{e}'_3, \mathbf{e}'_4\}$, where $\mathbf{e}'_1 = \mathbf{e}_1$, $\mathbf{e}'_2 = \mathbf{e}_1 + \mathbf{e}_2$, $\mathbf{e}'_3 = \mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3$, and $\mathbf{e}'_4 = \mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3 + \mathbf{e}_4$. Suppose that a linear transformation has a matrix representation in $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4\}$ of the form

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & 1 & 0 \\ -3 & 2 & 0 & 1 \\ 0 & 1 & -1 & 2 \\ 1 & 0 & 0 & 3 \end{bmatrix}.$$

Find the matrix representation of this linear transformation in the basis $\{\mathbf{e}'_1, \mathbf{e}'_2, \mathbf{e}'_3, \mathbf{e}'_4\}$.

3.5 Consider a linear transformation given by the matrix

$$\mathbf{A} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 2 & 5 & 2 & 1 \\ -1 & 1 & 0 & 3 \end{bmatrix}.$$

Find a basis for \mathbb{R}^4 with respect to which the matrix representation for the linear transformation above is diagonal.

3.6 Let $\lambda_1, \dots, \lambda_n$ be the eigenvalues of the matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$. Show that the eigenvalues of the matrix $\mathbf{I}_n - \mathbf{A}$ are $1 - \lambda_1, \dots, 1 - \lambda_n$.

3.7 Let \mathcal{V} be a subspace. Show that \mathcal{V}^\perp is also a subspace.

3.8 Find the nullspace of

$$\mathbf{A} = \begin{bmatrix} 4 & -2 & 0 \\ 2 & 1 & -1 \\ 2 & -3 & 1 \end{bmatrix}.$$

3.9 Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a matrix. Show that $\mathcal{R}(\mathbf{A})$ is a subspace of \mathbb{R}^m and $\mathcal{N}(\mathbf{A})$ is a subspace of \mathbb{R}^n .

3.10 Prove that if \mathbf{A} and \mathbf{B} are two matrices with m rows, and $\mathcal{N}(\mathbf{A}^\top) \subset \mathcal{N}(\mathbf{B}^\top)$, then $\mathcal{R}(\mathbf{B}) \subset \mathcal{R}(\mathbf{A})$.

Hint: Use the fact that for any matrix \mathbf{M} with m rows, we have $\dim \mathcal{R}(\mathbf{M}) + \dim \mathcal{N}(\mathbf{M}^\top) = m$ [this is one of the fundamental theorems of linear algebra (see [126, p. 75])].

3.11 Let \mathcal{V} be a subspace. Show that $(\mathcal{V}^\perp)^\perp = \mathcal{V}$.

Hint: Use Exercise 3.10.

3.12 Let \mathcal{V} and \mathcal{W} be subspaces. Show that if $\mathcal{V} \subset \mathcal{W}$, then $\mathcal{W}^\perp \subset \mathcal{V}^\perp$.

3.13 Let \mathcal{V} be a subspace of \mathbb{R}^n . Show that there exist matrices \mathbf{V} and \mathbf{U} such that $\mathcal{V} = \mathcal{R}(\mathbf{V}) = \mathcal{N}(\mathbf{U})$.

3.14 Let \mathbf{P} be an orthogonal projector onto a subspace \mathcal{V} . Show that

a. $\mathbf{P}\mathbf{x} = \mathbf{x}$ for all $\mathbf{x} \in \mathcal{V}$.

b. $\mathcal{R}(\mathbf{P}) = \mathcal{V}$.

3.15 Is the quadratic form

$$\mathbf{x}^\top \begin{bmatrix} 1 & -8 \\ 1 & 1 \end{bmatrix} \mathbf{x}$$

positive definite, positive semidefinite, negative definite, negative semidefinite, or indefinite?

3.16 Let

$$\mathbf{A} = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 0 \end{bmatrix}.$$

Show that although all leading principal minors of \mathbf{A} are nonnegative, \mathbf{A} is not positive semidefinite.

3.17 Consider the matrix

$$\mathbf{Q} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

- a. Is this matrix positive definite, negative definite, or indefinite?
- b. Is this matrix positive definite, negative definite, or indefinite on the subspace

$$\mathcal{M} = \{\mathbf{x} : x_1 + x_2 + x_3 = 0\} ?$$

3.18 For each of the following quadratic forms, determine if it is positive definite, negative definite, positive semidefinite, negative semidefinite, or indefinite.

- a. $f(x_1, x_2, x_3) = x_2^2$
- b. $f(x_1, x_2, x_3) = x_1^2 + 2x_2^2 - x_1x_3$
- c. $f(x_1, x_2, x_3) = x_1^2 + x_3^2 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3$

3.19 Find a transformation that brings the following quadratic form into the diagonal form:

$$f(x_1, x_2, x_3) = 4x_1^2 + x_2^2 + 9x_3^2 - 4x_1x_2 - 6x_2x_3 + 12x_1x_3.$$

Hint: Read the proof of Theorem 3.6.

3.20 Consider the quadratic form

$$f(x_1, x_2, x_3) = x_1^2 + x_2^2 + 5x_3^2 + 2\xi x_1 x_2 - 2x_1 x_3 + 4x_2 x_3.$$

Find the values of the parameter ξ for which this quadratic form is positive definite.

3.21 Consider the function $\langle \cdot, \cdot \rangle_Q : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$, defined by $\langle \mathbf{x}, \mathbf{y} \rangle_Q = \mathbf{x}^\top Q \mathbf{y}$, where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $Q \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix. Show that $\langle \cdot, \cdot \rangle_Q$ satisfies conditions 1 to 4 for inner products (see Section 2.4).

3.22 Consider the vector norm $\|\cdot\|_\infty$ on \mathbb{R}^n given by $\|\mathbf{x}\|_\infty = \max_i |x_i|$, where $\mathbf{x} = [x_1, \dots, x_n]^\top$. Define the norm $\|\cdot\|_\infty$ on \mathbb{R}^m similarly. Show that the matrix norm induced by these vector norms is given by

$$\|\mathbf{A}\|_\infty = \max_i \sum_{k=1}^n |a_{ik}|,$$

where a_{ij} is the (i, j) th element of $\mathbf{A} \in \mathbb{R}^{m \times n}$.

3.23 Consider the vector norm $\|\cdot\|_1$ on \mathbb{R}^n given by $\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i|$, where $\mathbf{x} = [x_1, \dots, x_n]^\top$. Define the norm $\|\cdot\|_1$ on \mathbb{R}^m similarly. Show that the matrix norm induced by these vector norms is given by

$$\|\mathbf{A}\|_1 = \max_k \sum_{i=1}^m |a_{ik}|,$$

where a_{ij} is the (i, j) th element of $\mathbf{A} \in \mathbb{R}^{m \times n}$.

CHAPTER 4

CONCEPTS FROM GEOMETRY

4.1 Line Segments

In the following analysis we concern ourselves only with \mathbb{R}^n . The elements of this space are the n -component vectors $\mathbf{x} = [x_1, x_2, \dots, x_n]^\top$.

The *line segment* between two points \mathbf{x} and \mathbf{y} in \mathbb{R}^n is the set of points on the straight line joining points \mathbf{x} and \mathbf{y} (see Figure 4.1). Note that if \mathbf{z} lies on the line segment between \mathbf{x} and \mathbf{y} , then

$$\mathbf{z} - \mathbf{y} = \alpha(\mathbf{x} - \mathbf{y}),$$

where α is a real number from the interval $[0, 1]$. The equation above can be rewritten as $\mathbf{z} = \alpha\mathbf{x} + (1 - \alpha)\mathbf{y}$. Hence, the line segment between \mathbf{x} and \mathbf{y} can be represented as

$$\{\alpha\mathbf{x} + (1 - \alpha)\mathbf{y} : \alpha \in [0, 1]\}.$$

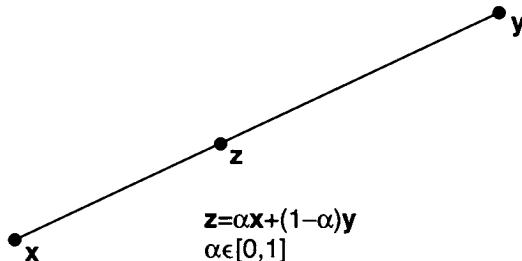


Figure 4.1 Line segment.

4.2 Hyperplanes and Linear Varieties

Let $u_1, u_2, \dots, u_n, v \in \mathbb{R}$, where at least one of the u_i is nonzero. The set of all points $\mathbf{x} = [x_1, x_2, \dots, x_n]^\top$ that satisfy the linear equation

$$u_1x_1 + u_2x_2 + \cdots + u_nx_n = v$$

is called a *hyperplane* of the space \mathbb{R}^n . We may describe the hyperplane by

$$\{\mathbf{x} \in \mathbb{R}^n : \mathbf{u}^\top \mathbf{x} = v\},$$

where

$$\mathbf{u} = [u_1, u_2, \dots, u_n]^\top.$$

A hyperplane is not necessarily a subspace of \mathbb{R}^n since, in general, it does not contain the origin. For $n = 2$, the equation of the hyperplane has the form $u_1x_1 + u_2x_2 = v$, which is the equation of a straight line. Thus, straight lines are hyperplanes in \mathbb{R}^2 . In \mathbb{R}^3 (three-dimensional space), hyperplanes are ordinary planes. By translating a hyperplane so that it contains the origin of \mathbb{R}^n , it becomes a subspace of \mathbb{R}^n (see Figure 4.2). Because the dimension of this subspace is $n - 1$, we say that the hyperplane has dimension $n - 1$.

The hyperplane $H = \{\mathbf{x} : u_1x_1 + \cdots + u_nx_n = v\}$ divides \mathbb{R}^n into two *half-spaces*. One of these half-spaces consists of the points satisfying the inequality $u_1x_1 + u_2x_2 + \cdots + u_nx_n \geq v$, denoted

$$H_+ = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{u}^\top \mathbf{x} \geq v\},$$

where, as before,

$$\mathbf{u} = [u_1, u_2, \dots, u_n]^\top.$$

The other half-space consists of the points satisfying the inequality $u_1x_1 + u_2x_2 + \cdots + u_nx_n \leq v$, denoted

$$H_- = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{u}^\top \mathbf{x} \leq v\}.$$

The half-space H_+ is called the *positive half-space*, and the half-space H_- is called the *negative half-space*.

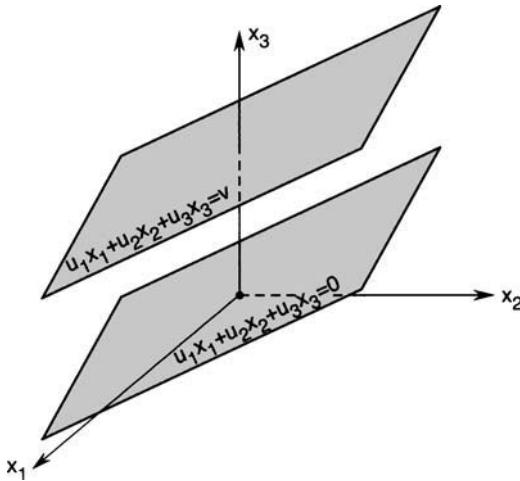


Figure 4.2 Translation of a hyperplane.

Let $\mathbf{a} = [a_1, a_2, \dots, a_n]^\top$ be an arbitrary point of the hyperplane H . Thus, $\mathbf{u}^\top \mathbf{a} - v = 0$. We can write

$$\begin{aligned}\mathbf{u}^\top \mathbf{x} - v &= \mathbf{u}^\top \mathbf{x} - v - (\mathbf{u}^\top \mathbf{a} - v) \\ &= \mathbf{u}^\top (\mathbf{x} - \mathbf{a}) \\ &= u_1(x_1 - a_1) + u_2(x_2 - a_2) + \cdots + u_n(x_n - a_n) = 0.\end{aligned}$$

The numbers $(x_i - a_i)$, $i = 1, \dots, n$, are the components of the vector $\mathbf{x} - \mathbf{a}$. Therefore, the hyperplane H consists of the points \mathbf{x} for which $\langle \mathbf{u}, \mathbf{x} - \mathbf{a} \rangle = 0$. In other words, the hyperplane H consists of the points \mathbf{x} for which the vectors \mathbf{u} and $\mathbf{x} - \mathbf{a}$ are orthogonal (see Figure 4.3). We call the vector \mathbf{u} the *normal* to the hyperplane H . The set H_+ consists of those points \mathbf{x} for which $\langle \mathbf{u}, \mathbf{x} - \mathbf{a} \rangle \geq 0$, and H_- consists of those points \mathbf{x} for which $\langle \mathbf{u}, \mathbf{x} - \mathbf{a} \rangle \leq 0$.

A *linear variety* is a set of the form

$$\{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b}\}$$

for some matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and vector $\mathbf{b} \in \mathbb{R}^m$. If $\dim \mathcal{N}(\mathbf{A}) = r$, we say that the linear variety has dimension r . A linear variety is a subspace if and only if $\mathbf{b} = \mathbf{0}$. If $\mathbf{A} = \mathbf{O}$, the linear variety is \mathbb{R}^n . If the dimension of the linear variety is less than n , then it is the intersection of a finite number of hyperplanes.

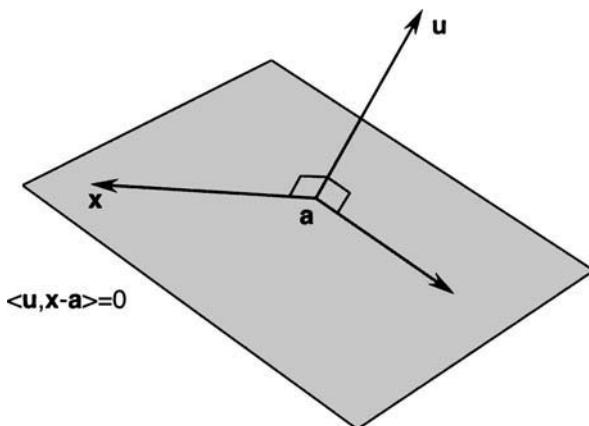


Figure 4.3 The hyperplane $H = \{x \in \mathbb{R}^n : u^\top (x - a) = 0\}$.

4.3 Convex Sets

Recall that the line segment between two points $u, v \in \mathbb{R}^n$ is the set $\{w \in \mathbb{R}^n : w = \alpha u + (1 - \alpha)v, \alpha \in [0, 1]\}$. A point $w = \alpha u + (1 - \alpha)v$ (where $\alpha \in [0, 1]$) is called a *convex combination* of the points u and v .

A set $\Theta \subset \mathbb{R}^n$ is *convex* if for all $u, v \in \Theta$, the line segment between u and v is in Θ . Figure 4.4 gives examples of convex sets, whereas Figure 4.5 gives examples of sets that are not convex. Note that Θ is convex if and only if $\alpha u + (1 - \alpha)v \in \Theta$ for all $u, v \in \Theta$ and $\alpha \in (0, 1)$.

Examples of convex sets include the following:

- The empty set
- A set consisting of a single point
- A line or a line segment

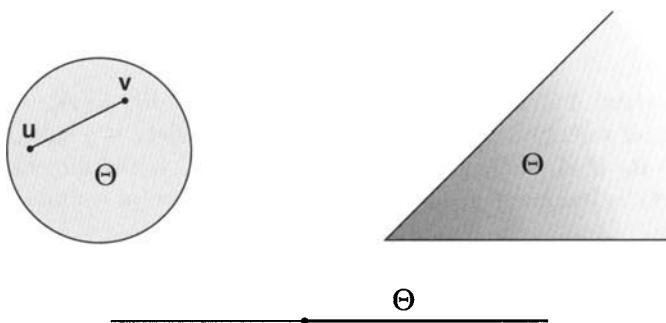


Figure 4.4 Convex sets.

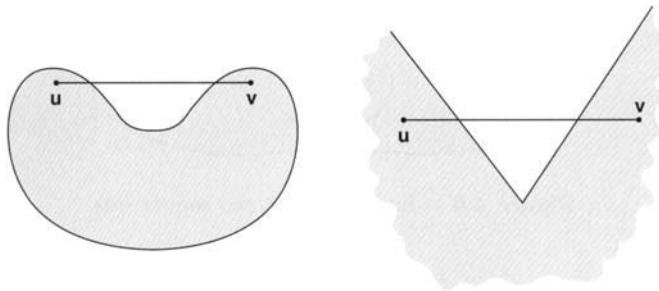


Figure 4.5 Sets that are not convex.

- A subspace
- A hyperplane
- A linear variety
- A half-space
- \mathbb{R}^n

Theorem 4.1 Convex subsets of \mathbb{R}^n have the following properties:

- a. If Θ is a convex set and β is a real number, then the set

$$\beta\Theta = \{\mathbf{x} : \mathbf{x} = \beta\mathbf{v}, \mathbf{v} \in \Theta\}$$

is also convex.

- b. If Θ_1 and Θ_2 are convex sets, then the set

$$\Theta_1 + \Theta_2 = \{\mathbf{x} : \mathbf{x} = \mathbf{v}_1 + \mathbf{v}_2, \mathbf{v}_1 \in \Theta_1, \mathbf{v}_2 \in \Theta_2\}$$

is also convex.

- c. The intersection of any collection of convex sets is convex (see Figure 4.6 for an illustration of this result for two sets). \square

Proof.

- a. Let $\beta\mathbf{v}_1, \beta\mathbf{v}_2 \in \beta\Theta$, where $\mathbf{v}_1, \mathbf{v}_2 \in \Theta$. Because Θ is convex, we have $\alpha\mathbf{v}_1 + (1 - \alpha)\mathbf{v}_2 \in \Theta$ for any $\alpha \in (0, 1)$. Hence,

$$\alpha\beta\mathbf{v}_1 + (1 - \alpha)\beta\mathbf{v}_2 = \beta(\alpha\mathbf{v}_1 + (1 - \alpha)\mathbf{v}_2) \in \beta\Theta,$$

and thus $\beta\Theta$ is convex.

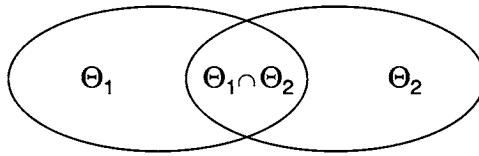


Figure 4.6 Intersection of two convex sets.

- b. Let $\mathbf{v}_1, \mathbf{v}_2 \in \Theta_1 + \Theta_2$. Then, $\mathbf{v}_1 = \mathbf{v}'_1 + \mathbf{v}''_1$, and $\mathbf{v}_2 = \mathbf{v}'_2 + \mathbf{v}''_2$, where $\mathbf{v}'_1, \mathbf{v}'_2 \in \Theta_1$, and $\mathbf{v}''_1, \mathbf{v}''_2 \in \Theta_2$. Because Θ_1 and Θ_2 are convex, for all $\alpha \in (0, 1)$,

$$\mathbf{x}_1 = \alpha \mathbf{v}'_1 + (1 - \alpha) \mathbf{v}'_2 \in \Theta_1$$

and

$$\mathbf{x}_2 = \alpha \mathbf{v}''_1 + (1 - \alpha) \mathbf{v}''_2 \in \Theta_2.$$

By definition of $\Theta_1 + \Theta_2$, $\mathbf{x}_1 + \mathbf{x}_2 \in \Theta_1 + \Theta_2$. Now,

$$\begin{aligned}\alpha \mathbf{v}_1 + (1 - \alpha) \mathbf{v}_2 &= \alpha(\mathbf{v}'_1 + \mathbf{v}''_1) + (1 - \alpha)(\mathbf{v}'_2 + \mathbf{v}''_2) \\ &= \mathbf{x}_1 + \mathbf{x}_2 \in \Theta_1 + \Theta_2.\end{aligned}$$

Hence, $\Theta_1 + \Theta_2$ is convex.

- c. Let C be a collection of convex sets. Let $\mathbf{x}_1, \mathbf{x}_2 \in \bigcap_{\Theta \in C} \Theta$ (where $\bigcap_{\Theta \in C} \Theta$ represents the intersection of all elements in C). Then, $\mathbf{x}_1, \mathbf{x}_2 \in \Theta$ for each $\Theta \in C$. Because each $\Theta \in C$ is convex, $\alpha \mathbf{x}_1 + (1 - \alpha) \mathbf{x}_2 \in \Theta$ for all $\alpha \in (0, 1)$ and each $\Theta \in C$. Thus, $\alpha \mathbf{x}_1 + (1 - \alpha) \mathbf{x}_2 \in \bigcap_{\Theta \in C} \Theta$. ■

A point \mathbf{x} in a convex set Θ is said to be an *extreme point* of Θ if there are no two distinct points \mathbf{u} and \mathbf{v} in Θ such that $\mathbf{x} = \alpha \mathbf{u} + (1 - \alpha) \mathbf{v}$ for some $\alpha \in (0, 1)$. For example, in Figure 4.4, any point on the boundary of the disk is an extreme point, the vertex (corner) of the set on the right is an extreme point, and the endpoint of the half-line is also an extreme point.

4.4 Neighborhoods

A *neighborhood* of a point $\mathbf{x} \in \mathbb{R}^n$ is the set

$$\{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y} - \mathbf{x}\| < \varepsilon\},$$

where ε is some positive number. The neighborhood is also called a *ball* with radius ε and center \mathbf{x} .

In the plane \mathbb{R}^2 , a neighborhood of $\mathbf{x} = [x_1, x_2]^\top$ consists of all the points inside a disk centered at \mathbf{x} . In \mathbb{R}^3 , a neighborhood of $\mathbf{x} = [x_1, x_2, x_3]^\top$ consists of all the points inside a sphere centered at \mathbf{x} (see Figure 4.7).

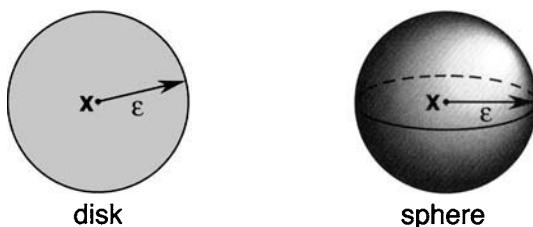


Figure 4.7 Examples of neighborhoods of a point in \mathbb{R}^2 and \mathbb{R}^3 .

A point $x \in S$ is said to be an *interior point* of the set S if the set S contains some neighborhood of x ; that is, if all points within some neighborhood of x are also in S (see Figure 4.8). The set of all the interior points of S is called the *interior* of S .

A point x is said to be a *boundary point* of the set S if every neighborhood of x contains a point in S and a point not in S (see Figure 4.8). Note that a boundary point of S may or may not be an element of S . The set of all boundary points of S is called the *boundary* of S .

A set S is said to be *open* if it contains a neighborhood of each of its points; that is, if each of its points is an interior point, or equivalently, if S contains no boundary points.

A set S is said to be *closed* if it contains its boundary (see Figure 4.9). We can show that a set is closed if and only if its complement is open.

A set that is contained in a ball of finite radius is said to be *bounded*. A set is *compact* if it is both closed and bounded. Compact sets are important in optimization problems for the following reason.

Theorem 4.2 Theorem of Weierstrass. Let $f : \Omega \rightarrow \mathbb{R}$ be a continuous function, where $\Omega \subset \mathbb{R}^n$ is a compact set. Then, there exists a point $x_0 \in \Omega$

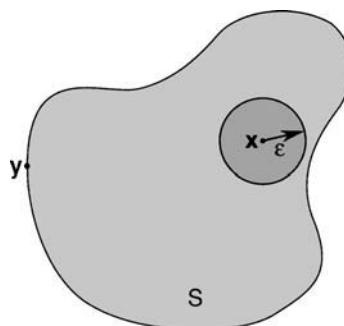


Figure 4.8 x is an interior point; y is a boundary point.

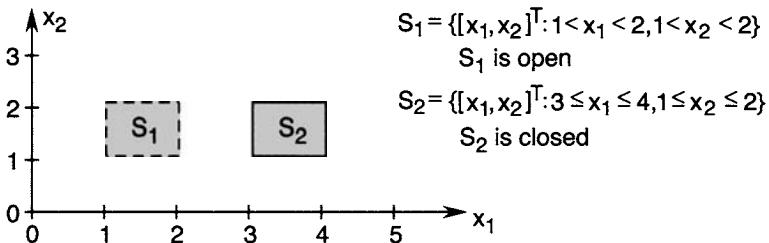


Figure 4.9 Open and closed sets.

such that $f(\mathbf{x}_0) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \Omega$. In other words, f achieves its minimum on Ω . \square

Proof. See [112, p. 89] or [2, p. 154]. \blacksquare

4.5 Polytopes and Polyhedra

Let Θ be a convex set, and suppose that \mathbf{y} is a boundary point of Θ . A hyperplane passing through \mathbf{y} is called a *hyperplane of support* (or *supporting hyperplane*) of the set Θ if the entire set Θ lies completely in one of the two half-spaces into which this hyperplane divides the space \mathbb{R}^n .

Recall that by Theorem 4.1, the intersection of any number of convex sets is convex. In what follows we are concerned with the intersection of a finite number of half-spaces. Because every half-space H_+ or H_- is convex in \mathbb{R}^n , the intersection of any number of half-spaces is a convex set.

A set that can be expressed as the intersection of a finite number of half-spaces is called a *convex polytope* (see Figure 4.10).

A nonempty bounded polytope is called a *polyhedron* (see Figure 4.11).

For every convex polyhedron $\Theta \subset \mathbb{R}^n$, there exists a nonnegative integer $k \leq n$ such that Θ is contained in a linear variety of dimension k , but is not

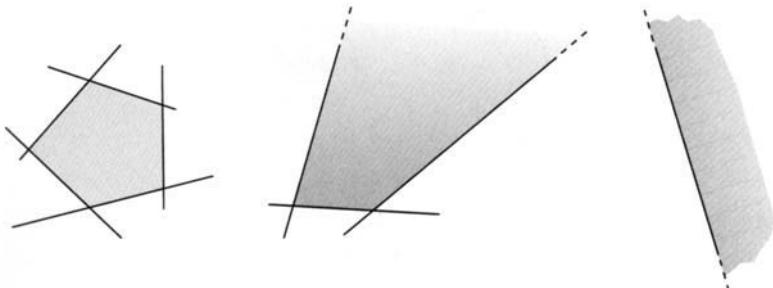


Figure 4.10 Polytopes.

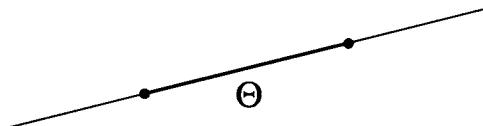


Figure 4.11 One-dimensional polyhedron.

entirely contained in any $(k - 1)$ -dimensional linear variety of \mathbb{R}^n . Furthermore, there exists only one k -dimensional linear variety containing Θ , called the *carrier* of the polyhedron Θ , and k is called the dimension of Θ . For example, a zero-dimensional polyhedron is a point of \mathbb{R}^n , and its carrier is itself. A one-dimensional polyhedron is a segment, and its carrier is the straight line on which it lies. The boundary of any k -dimensional polyhedron, $k > 0$, consists of a finite number of $(k - 1)$ -dimensional polyhedra. For example, the boundary of a one-dimensional polyhedron consists of two points that are the endpoints of the segment.

The $(k - 1)$ -dimensional polyhedra forming the boundary of a k -dimensional polyhedron are called the *faces* of the polyhedron. Each of these faces has, in turn, $(k - 2)$ -dimensional faces. We also consider each of these $(k - 2)$ -dimensional faces to be faces of the original k -dimensional polyhedron. Thus, every k -dimensional polyhedron has faces of dimensions $k - 1, k - 2, \dots, 1, 0$. A zero-dimensional face of a polyhedron is called a *vertex*, and a one-dimensional face is called an *edge*.

EXERCISES

4.1 Show that a set $S \subset \mathbb{R}^n$ is a linear variety if and only if for all $\mathbf{x}, \mathbf{y} \in S$ and $\alpha \in \mathbb{R}$, we have $\alpha\mathbf{x} + (1 - \alpha)\mathbf{y} \in S$.

4.2 Show that the set $\{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| \leq r\}$ is convex, where $r > 0$ is a given real number and $\|\mathbf{x}\| = \sqrt{\mathbf{x}^\top \mathbf{x}}$ is the Euclidean norm of $\mathbf{x} \in \mathbb{R}^n$.

4.3 Show that for any matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and vector $\mathbf{b} \in \mathbb{R}^m$, the set (linear variety) $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{b}\}$ is convex.

4.4 Show that the set $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq \mathbf{0}\}$ is convex (where $\mathbf{x} \geq \mathbf{0}$ means that every component of \mathbf{x} is nonnegative).

CHAPTER 5

ELEMENTS OF CALCULUS

5.1 Sequences and Limits

A *sequence of real numbers* is a function whose domain is the set of natural numbers $1, 2, \dots, k, \dots$ and whose range is contained in \mathbb{R} . Thus, a sequence of real numbers can be viewed as a set of numbers $\{x_1, x_2, \dots, x_k, \dots\}$, which is often also denoted as $\{x_k\}$ (or sometimes as $\{x_k\}_{k=1}^{\infty}$, to indicate explicitly the range of values that k can take).

A sequence $\{x_k\}$ is *increasing* if $x_1 < x_2 < \dots < x_k \dots$; that is, $x_k < x_{k+1}$ for all k . If $x_k \leq x_{k+1}$, then we say that the sequence is *nondecreasing*. Similarly, we can define *decreasing* and *nonincreasing sequences*. Nonincreasing or nondecreasing sequences are called *monotone sequences*.

A number $x^* \in \mathbb{R}$ is called the *limit* of the sequence $\{x_k\}$ if for any positive ε there is a number K (which may depend on ε) such that for all $k > K$, $|x_k - x^*| < \varepsilon$; that is, x_k lies between $x^* - \varepsilon$ and $x^* + \varepsilon$ for all $k > K$. In this case we write

$$x^* = \lim_{k \rightarrow \infty} x_k$$

or

$$x_k \rightarrow x^*.$$

A sequence that has a limit is called a *convergent sequence*.

The notion of a sequence can be extended to sequences with elements in \mathbb{R}^n . Specifically, a sequence in \mathbb{R}^n is a function whose domain is the set of natural numbers $1, 2, \dots, k, \dots$ and whose range is contained in \mathbb{R}^n . We use the notation $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots\}$ or $\{\mathbf{x}^{(k)}\}$ for sequences in \mathbb{R}^n . For limits of sequences in \mathbb{R}^n , we need to replace absolute values with vector norms. In other words, \mathbf{x}^* is the limit of $\{\mathbf{x}^{(k)}\}$ if for any positive ε there is a number K (which may depend on ε) such that for all $k > K$, $\|\mathbf{x}^{(k)} - \mathbf{x}^*\| < \varepsilon$. As before, if a sequence $\{\mathbf{x}^{(k)}\}$ is convergent, we write $\mathbf{x}^* = \lim_{k \rightarrow \infty} \mathbf{x}^{(k)}$ or $\mathbf{x}^{(k)} \rightarrow \mathbf{x}^*$.

Theorem 5.1 *A convergent sequence has only one limit.*

□

Proof. We prove this result by contradiction. Suppose that a sequence $\{\mathbf{x}^{(k)}\}$ has two different limits, say \mathbf{x}_1 and \mathbf{x}_2 . Then, we have $\|\mathbf{x}_1 - \mathbf{x}_2\| > 0$. Let

$$\varepsilon = \frac{1}{2} \|\mathbf{x}_1 - \mathbf{x}_2\|.$$

From the definition of a limit, there exist K_1 and K_2 such that for $k > K_1$ we have $\|\mathbf{x}^{(k)} - \mathbf{x}_1\| < \varepsilon$, and for $k > K_2$ we have $\|\mathbf{x}^{(k)} - \mathbf{x}_2\| < \varepsilon$. Let $K = \max\{K_1, K_2\}$. Then, if $k > K$, we have $\|\mathbf{x}^{(k)} - \mathbf{x}_1\| < \varepsilon$ and $\|\mathbf{x}^{(k)} - \mathbf{x}_2\| < \varepsilon$. Adding $\|\mathbf{x}^{(k)} - \mathbf{x}_1\| < \varepsilon$ and $\|\mathbf{x}^{(k)} - \mathbf{x}_2\| < \varepsilon$ yields

$$\|\mathbf{x}^{(k)} - \mathbf{x}_1\| + \|\mathbf{x}^{(k)} - \mathbf{x}_2\| < 2\varepsilon.$$

Applying the triangle inequality gives

$$\begin{aligned}\|\mathbf{x}_1 - \mathbf{x}_2\| &= \|\mathbf{x}^{(k)} - \mathbf{x}_1 - (\mathbf{x}^{(k)} - \mathbf{x}_2)\| \\ &= \|(\mathbf{x}^{(k)} - \mathbf{x}_1) - (\mathbf{x}^{(k)} - \mathbf{x}_2)\| \\ &\leq \|\mathbf{x}^{(k)} - \mathbf{x}_1\| + \|\mathbf{x}^{(k)} - \mathbf{x}_2\|.\end{aligned}$$

Therefore,

$$\|\mathbf{x}_1 - \mathbf{x}_2\| = \|\mathbf{x}_1 - \mathbf{x}_2\| < 2\varepsilon.$$

However, this contradicts the assumption that $\|\mathbf{x}_1 - \mathbf{x}_2\| = 2\varepsilon$, which completes the proof. ■

A sequence $\{\mathbf{x}^{(k)}\}$ in \mathbb{R}^n is *bounded* if there exists a number $B \geq 0$ such that $\|\mathbf{x}^{(k)}\| \leq B$ for all $k = 1, 2, \dots$.

Theorem 5.2 *Every convergent sequence is bounded.*

□

Proof. Let $\{\mathbf{x}^{(k)}\}$ be a convergent sequence with limit \mathbf{x}^* . Choose $\varepsilon = 1$. Then, by definition of the limit, there exists a natural number K such that for all $k > K$,

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\| < 1.$$

By the result of Exercise 2.9, we get

$$\|\mathbf{x}^{(k)}\| - \|\mathbf{x}^*\| \leq \|\mathbf{x}^{(k)} - \mathbf{x}^*\| < 1 \quad \text{for all } k > K.$$

Therefore,

$$\|\mathbf{x}^{(k)}\| < \|\mathbf{x}^*\| + 1 \quad \text{for all } k > K.$$

Letting

$$B = \max \left\{ \|\mathbf{x}^{(1)}\|, \|\mathbf{x}^{(2)}\|, \dots, \|\mathbf{x}^{(K)}\|, \|\mathbf{x}^*\| + 1 \right\},$$

we have

$$B \geq \|\mathbf{x}^{(k)}\| \quad \text{for all } k,$$

which means that the sequence $\{\mathbf{x}^{(k)}\}$ is bounded. ■

For a sequence $\{x_k\}$ in \mathbb{R} , a number B is called an *upper bound* if $x_k \leq B$ for all $k = 1, 2, \dots$. In this case, we say that $\{x_k\}$ is *bounded above*. Similarly, B is called a *lower bound* if $x_k \geq B$ for all $k = 1, 2, \dots$. In this case, we say that $\{x_k\}$ is *bounded below*. Clearly, a sequence is bounded if it is both bounded above and bounded below.

Any sequence $\{x_k\}$ in \mathbb{R} that has an upper bound has a *least upper bound* (also called the *supremum*), which is the smallest number B that is an upper bound of $\{x_k\}$. Similarly, any sequence $\{x_k\}$ in \mathbb{R} that has a lower bound has a *greatest lower bound* (also called the *infimum*). If B is the least upper bound of the sequence $\{x_k\}$, then $x_k \leq B$ for all k , and for any $\varepsilon > 0$, there exists a number K such that $x_K > B - \varepsilon$. An analogous statement applies to the greatest lower bound: If B is the greatest lower bound of $\{x_k\}$, then $x_k \geq B$ for all k , and for any $\varepsilon > 0$, there exists a number K such that $x_K < B + \varepsilon$.

Theorem 5.3 Every monotone bounded sequence in \mathbb{R} is convergent. □

Proof. We prove the theorem for nondecreasing sequences. The proof for nonincreasing sequences is analogous.

Let $\{x_k\}$ be a bounded nondecreasing sequence in \mathbb{R} and x^* the least upper bound. Fix a number $\varepsilon > 0$. Then, there exists a number K such that $x_K > x^* - \varepsilon$. Because $\{x_k\}$ is nondecreasing, for any $k \geq K$,

$$x_k \geq x_K > x^* - \varepsilon.$$

Also, because x^* is an upper bound of $\{x_k\}$, we have

$$x_k \leq x^* < x^* + \varepsilon.$$

Therefore, for any $k \geq K$,

$$|x_k - x^*| < \varepsilon,$$

which means that $x_k \rightarrow x^*$. ■

Suppose that we are given a sequence $\{\mathbf{x}^{(k)}\}$ and an increasing sequence of natural numbers $\{m_k\}$. The sequence

$$\{\mathbf{x}^{(m_k)}\} = \{\mathbf{x}^{(m_1)}, \mathbf{x}^{(m_2)}, \dots\}$$

is called a *subsequence* of the sequence $\{\mathbf{x}^{(k)}\}$. A subsequence of a given sequence can thus be obtained by neglecting some elements of the given sequence.

Theorem 5.4 Consider a convergent sequence $\{\mathbf{x}^{(k)}\}$ with limit \mathbf{x}^* . Then, any subsequence of $\{\mathbf{x}^{(k)}\}$ also converges to \mathbf{x}^* . □

Proof. Let $\{\mathbf{x}^{(m_k)}\}$ be a subsequence of $\{\mathbf{x}^{(k)}\}$, where $\{m_k\}$ is an increasing sequence of natural numbers. Observe that $m_k \geq k$ for all $k = 1, 2, \dots$. To show this, first note that $m_1 \geq 1$ because m_1 is a natural number. Next, we proceed by induction by assuming that $m_k \geq k$. Then, we have $m_{k+1} > m_k \geq k$, which implies that $m_{k+1} \geq k + 1$. Therefore, we have shown that $m_k \geq k$ for all $k = 1, 2, \dots$

Let $\varepsilon > 0$ be given. Then, by definition of the limit, there exists K such that $\|\mathbf{x}^{(k)} - \mathbf{x}^*\| < \varepsilon$ for any $k > K$. Because $m_k \geq k$, we also have $\|\mathbf{x}^{(m_k)} - \mathbf{x}^*\| < \varepsilon$ for any $k > K$. This means that

$$\lim_{k \rightarrow \infty} \mathbf{x}^{(m_k)} = \mathbf{x}^*. \quad \blacksquare$$

It turns out that any bounded sequence contains a convergent subsequence. This result is called the *Bolzano-Weierstrass theorem* (see [2, p. 70]).

Consider a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and a point $\mathbf{x}_0 \in \mathbb{R}^n$. Suppose that there exists \mathbf{f}^* such that for any convergent sequence $\{\mathbf{x}^{(k)}\}$ with limit \mathbf{x}_0 , we have

$$\lim_{k \rightarrow \infty} \mathbf{f}(\mathbf{x}^{(k)}) = \mathbf{f}^*.$$

Then, we use the notation

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}_0} \mathbf{f}(\mathbf{x})$$

to represent the limit \mathbf{f}^* .

It turns out that \mathbf{f} is continuous at \mathbf{x}_0 if and only if for any convergent sequence $\{\mathbf{x}^{(k)}\}$ with limit \mathbf{x}_0 , we have

$$\lim_{k \rightarrow \infty} \mathbf{f}(\mathbf{x}^{(k)}) = \mathbf{f}\left(\lim_{k \rightarrow \infty} \mathbf{x}^{(k)}\right) = \mathbf{f}(\mathbf{x}_0)$$

(see [2, p. 137]). Therefore, using the notation introduced above, the function \mathbf{f} is continuous at \mathbf{x}_0 if and only if

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}_0} \mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_0).$$

We end this section with some results involving sequences and limits of matrices. These results are useful in the analysis of algorithms (e.g., Newton's algorithm in Chapter 9).

We say that a sequence $\{\mathbf{A}_k\}$ of $m \times n$ matrices converges to the $m \times n$ matrix \mathbf{A} if

$$\lim_{k \rightarrow \infty} \|\mathbf{A} - \mathbf{A}_k\| = 0.$$

Lemma 5.1 Let $\mathbf{A} \in \mathbb{R}^{n \times n}$. Then, $\lim_{k \rightarrow \infty} \mathbf{A}^k = \mathbf{O}$ if and only if the eigenvalues of \mathbf{A} satisfy $|\lambda_i(\mathbf{A})| < 1$, $i = 1, \dots, n$. \square

Proof. To prove this theorem, we use the *Jordan form* (see, e.g., [47]). Specifically, it is well known that any square matrix is similar to the Jordan form: There exists a nonsingular \mathbf{T} such that

$$\mathbf{T}\mathbf{A}\mathbf{T}^{-1} = \text{diag} [\mathbf{J}_{m_1}(\lambda_1), \dots, \mathbf{J}_{m_s}(\lambda_1), \mathbf{J}_{n_1}(\lambda_2), \dots, \mathbf{J}_{t_v}(\lambda_q)] \triangleq \mathbf{J},$$

where $\mathbf{J}_r(\lambda)$ is the $r \times r$ matrix:

$$\mathbf{J}_r(\lambda) = \begin{bmatrix} \lambda & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & \lambda \end{bmatrix} \xrightarrow{-1 \text{ (multiple)}} \begin{pmatrix} -1 & 0 & 0 \\ 0 & -2 & 1 \\ 0 & 0 & -2 \end{pmatrix} = \text{diag} (\mathbf{J}(-1), \mathbf{J}(-2))$$

The $\lambda_1, \dots, \lambda_q$ above are distinct eigenvalues of \mathbf{A} , the multiplicity of λ_1 is $m_1 + \dots + m_s$, and so on.

We may rewrite the above as $\mathbf{A} = \mathbf{T}^{-1}\mathbf{J}\mathbf{T}$. To complete the proof, observe that

$$(\mathbf{J}_r(\lambda))^k = \begin{bmatrix} \lambda^k & \binom{k}{k-1} \lambda^{k-1} & \cdots & \binom{k}{k-r+1} \lambda^{k-r+1} \\ 0 & \lambda^k & \cdots & \binom{k}{k-r+2} \lambda^{k-r+2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda^k \end{bmatrix},$$

where

$$\binom{k}{i} = \frac{k!}{i!(k-i)!}.$$

Furthermore,

$$\mathbf{A}^k = \mathbf{T}^{-1}\mathbf{J}^k\mathbf{T}.$$

Hence,

$$\lim_{k \rightarrow \infty} \mathbf{A}^k = \mathbf{T}^{-1} \left(\lim_{k \rightarrow \infty} \mathbf{J}^k \right) \mathbf{T} = \mathbf{O}$$

if and only if $|\lambda_i| < 1$, $i = 1, \dots, n$.

Lemma 5.2 *The series of $n \times n$ matrices*

$$\boxed{I_n + A + A^2 + \cdots + A^k + \cdots} = (I_n - A)^{-1}$$

converges if and only if $\lim_{k \rightarrow \infty} A^k = \mathbf{O}$. In this case the sum of the series equals $(I_n - A)^{-1}$. \square

Proof. The necessity of the condition is obvious.

To prove the sufficiency, suppose that $\lim_{k \rightarrow \infty} A^k = \mathbf{O}$. By Lemma 5.1 we deduce that $|\lambda_i(A)| < 1$, $i = 1, \dots, n$. This implies that $\det(I_n - A) \neq 0$, and hence $(I_n - A)^{-1}$ exists. Consider now the following relation:

$$(I_n + A + A^2 + \cdots + A^k)(I_n - A) = I_n - A^{k+1}.$$

Postmultiplying the equation above by $(I_n - A)^{-1}$ yields

$$I_n + A + A^2 + \cdots + A^k = (I_n - A)^{-1} - A^{k+1}(I_n - A)^{-1}.$$

Hence,

$$\lim_{k \rightarrow \infty} \sum_{j=0}^k A^j = (I_n - A)^{-1},$$

because $\lim_{k \rightarrow \infty} A^{k+1} = \mathbf{O}$. Thus,

$$\sum_{j=0}^{\infty} A^j = (I_n - A)^{-1},$$

which completes the proof. \blacksquare

A matrix-valued function $A : \mathbb{R}^r \rightarrow \mathbb{R}^{n \times n}$ is continuous at a point $\xi_0 \in \mathbb{R}^r$ if

$$\text{Jednačina: } \lim_{\|\xi - \xi_0\| \rightarrow 0} \|A(\xi) - A(\xi_0)\| = 0.$$

Lemma 5.3 Let $A : \mathbb{R}^r \rightarrow \mathbb{R}^{n \times n}$ be an $n \times n$ matrix-valued function that is continuous at ξ_0 . If $A(\xi_0)^{-1}$ exists, then $A(\xi)^{-1}$ exists for ξ sufficiently close to ξ_0 and $A(\cdot)^{-1}$ is continuous at ξ_0 . \square

Proof. We follow [114]. We first prove the existence of $A(\xi)^{-1}$ for all ξ sufficiently close to ξ_0 . We have

$$A(\xi) = A(\xi_0) - A(\xi_0) + A(\xi) = A(\xi_0)(I_n - K(\xi)),$$

where

$$K(\xi) = A(\xi_0)^{-1}(A(\xi_0) - A(\xi)).$$

Thus,

$$\|\mathbf{K}(\boldsymbol{\xi})\| \leq \|\mathbf{A}(\boldsymbol{\xi}_0)^{-1}\| \|\mathbf{A}(\boldsymbol{\xi}_0) - \mathbf{A}(\boldsymbol{\xi})\|$$

and

$$\lim_{\|\boldsymbol{\xi} - \boldsymbol{\xi}_0\| \rightarrow 0} \|\mathbf{K}(\boldsymbol{\xi})\| = 0.$$

Because \mathbf{A} is continuous at $\boldsymbol{\xi}_0$, for all $\boldsymbol{\xi}$ close enough to $\boldsymbol{\xi}_0$, we have

$$\|\mathbf{A}(\boldsymbol{\xi}_0) - \mathbf{A}(\boldsymbol{\xi})\| \leq \frac{\theta}{\|\mathbf{A}(\boldsymbol{\xi}_0)^{-1}\|},$$

where $\theta \in (0, 1)$. Then,

$$\|\mathbf{K}(\boldsymbol{\xi})\| \leq \theta < 1$$

and

$$(\mathbf{I}_n - \mathbf{K}(\boldsymbol{\xi}))^{-1}$$

exists. But then

$$\mathbf{A}(\boldsymbol{\xi})^{-1} = (\mathbf{A}(\boldsymbol{\xi}_0)(\mathbf{I}_n - \mathbf{K}(\boldsymbol{\xi})))^{-1} = (\mathbf{I}_n - \mathbf{K}(\boldsymbol{\xi}))^{-1} \mathbf{A}(\boldsymbol{\xi}_0)^{-1},$$

which means that $\mathbf{A}(\boldsymbol{\xi})^{-1}$ exists for $\boldsymbol{\xi}$ sufficiently close to $\boldsymbol{\xi}_0$.

To prove the continuity of $\mathbf{A}(\cdot)^{-1}$ note that

$$\begin{aligned} \|\mathbf{A}(\boldsymbol{\xi}_0)^{-1} - \mathbf{A}(\boldsymbol{\xi})^{-1}\| &= \|\mathbf{A}(\boldsymbol{\xi})^{-1} - \mathbf{A}(\boldsymbol{\xi}_0)^{-1}\| \\ &= \|((\mathbf{I}_n - \mathbf{K}(\boldsymbol{\xi}))^{-1} - \mathbf{I}_n)\mathbf{A}(\boldsymbol{\xi}_0)^{-1}\|. \end{aligned}$$

However, since $\|\mathbf{K}(\boldsymbol{\xi})\| < 1$, it follows from Lemma 5.2 that

$$(\mathbf{I}_n - \mathbf{K}(\boldsymbol{\xi}))^{-1} - \mathbf{I}_n = \mathbf{K}(\boldsymbol{\xi}) + \mathbf{K}^2(\boldsymbol{\xi}) + \cdots = \mathbf{K}(\boldsymbol{\xi})(\mathbf{I}_n + \mathbf{K}(\boldsymbol{\xi}) + \cdots).$$

Hence,

$$\begin{aligned} \|(\mathbf{I}_n - \mathbf{K}(\boldsymbol{\xi}))^{-1} - \mathbf{I}_n\| &\leq \|\mathbf{K}(\boldsymbol{\xi})\|(1 + \|\mathbf{K}(\boldsymbol{\xi})\| + \|\mathbf{K}(\boldsymbol{\xi})\|^2 + \cdots) \\ &= \frac{\|\mathbf{K}(\boldsymbol{\xi})\|}{1 - \|\mathbf{K}(\boldsymbol{\xi})\|}, \end{aligned}$$

when $\|\mathbf{K}(\boldsymbol{\xi})\| < 1$. Therefore,

$$\|\mathbf{A}(\boldsymbol{\xi})^{-1} - \mathbf{A}(\boldsymbol{\xi}_0)^{-1}\| \leq \frac{\|\mathbf{K}(\boldsymbol{\xi})\|}{1 - \|\mathbf{K}(\boldsymbol{\xi})\|} \|\mathbf{A}(\boldsymbol{\xi}_0)^{-1}\|.$$

Because

$$\lim_{\|\boldsymbol{\xi} - \boldsymbol{\xi}_0\| \rightarrow 0} \|\mathbf{K}(\boldsymbol{\xi})\| = 0,$$

we obtain

$$\lim_{\|\boldsymbol{\xi} - \boldsymbol{\xi}_0\| \rightarrow 0} \|\mathbf{A}(\boldsymbol{\xi})^{-1} - \mathbf{A}(\boldsymbol{\xi}_0)^{-1}\| = 0,$$

which completes the proof. ■



ELEMENTS OF CALCULUS

5.2 Differentiability

$$|f(x) - ax_0 - b| < \epsilon$$

$$a = f'(x_0)$$

$y = ax + b$ $b \in \mathbb{R}$

$$L(x) = ax$$

Differential calculus is based on the idea of approximating an arbitrary function by an *affine function*. A function $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is *affine* if there exists a *linear function* $L : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and a vector $y \in \mathbb{R}^m$ such that

$$A(x) = L(x) + y$$

for every $x \in \mathbb{R}^n$. Consider a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and a point $x_0 \in \mathbb{R}^n$. We wish to find an affine function A that approximates f near the point x_0 . First, it is natural to impose the condition

$$A(x_0) = f(x_0).$$

Because $A(x) = L(x) + y$, we obtain $y = f(x_0) - L(x_0)$. By the linearity of L ,

$$L(x) + y = L(x) - L(x_0) + f(x_0) = L(x - x_0) + f(x_0).$$

Hence, we may write

$$A(x) = L(x - x_0) + f(x_0).$$

Next, we require that $A(x)$ approaches $f(x)$ faster than x approaches x_0 ; that is,

$$\lim_{x \rightarrow x_0, x \in \Omega} \frac{\|f(x) - A(x)\|}{\|x - x_0\|} = 0.$$

The conditions above on A ensure that A approximates f near x_0 in the sense that the error in the approximation at a given point is “small” compared with the distance of the point from x_0 .

In summary, a function $f : \Omega \rightarrow \mathbb{R}^m$, $\Omega \subset \mathbb{R}^n$, is said to be *differentiable* at $x_0 \in \Omega$ if there is an affine function that approximates f near x_0 ; that is, there exists a linear function $L : \mathbb{R}^n \rightarrow \mathbb{R}^m$ such that

$$\lim_{x \rightarrow x_0, x \in \Omega} \frac{\|f(x) - (L(x - x_0) + f(x_0))\|}{\|x - x_0\|} = 0.$$

The linear function L above is determined uniquely by f and x_0 and is called the *derivative of f at x_0* . The function f is said to be *differentiable* on Ω if f is differentiable at every point of its domain Ω .

In \mathbb{R} , an affine function has the form $ax + b$, with $a, b \in \mathbb{R}$. Hence, a real-valued function $f(x)$ of a real variable x that is differentiable at x_0 can be approximated near x_0 by a function

$$A(x) = ax + b.$$

Because $f(x_0) = A(x_0) = ax_0 + b$, we obtain

$$A(x) = ax + b = a(x - x_0) + f(x_0).$$

$a := f'(x_0)$

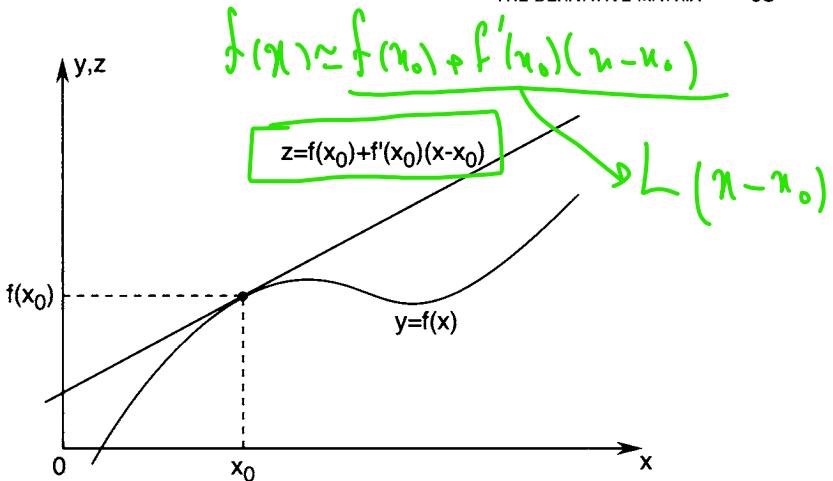


Figure 5.1 Illustration of the notion of the derivative.

The linear part of $\mathcal{A}(x)$, denoted earlier by $\mathcal{L}(x)$, is in this case just ax . The norm of a real number is its absolute value, so by the definition of differentiability we have

$$\lim_{x \rightarrow x_0} \frac{|f(x) - (a(x - x_0) + f(x_0))|}{|x - x_0|} = 0,$$

which is equivalent to

$$\lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0} = a.$$

The number a is commonly denoted $f'(x_0)$ and is called the derivative of f at x_0 . The affine function \mathcal{A} is therefore given by

$$\mathcal{A}(x) = f(x_0) + f'(x_0)(x - x_0).$$

This affine function is tangent to f at x_0 (see Figure 5.1).

5.3 The Derivative Matrix

Any linear transformation from \mathbb{R}^n to \mathbb{R}^m , and in particular the derivative \mathcal{L} of $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, can be represented by an $m \times n$ matrix. To find the matrix representation \mathbf{L} of the derivative \mathcal{L} of a differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, we use the natural basis $\{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ for \mathbb{R}^n . Consider the vectors

$$\mathbf{x}_j = \mathbf{x}_0 + t\mathbf{e}_j, \quad j = 1, \dots, n.$$

By the definition of the derivative, we have

$$\lim_{t \rightarrow 0} \frac{f(\mathbf{x}_j) - (t\mathbf{L}\mathbf{e}_j + f(\mathbf{x}_0))}{t} = 0$$

for $j = 1, \dots, n$. This means that

$$\lim_{t \rightarrow 0} \frac{f(x_j) - f(x_0)}{t} = L e_j$$

نظامیہ
عمرانیہ
ارٹ

for $j = 1, \dots, n$. But $\mathbf{L}e_j$ is the j th column of the matrix \mathbf{L} . On the other hand, the vector \mathbf{x}_j differs from \mathbf{x}_0 only in the j th coordinate, and in that coordinate the difference is just the number t . Therefore, the left side of the preceding equation is the partial derivative

$$\frac{\partial f}{\partial x_j}(x_0).$$

Because vector limits are computed by taking the limit of each coordinate function, it follows that if

$$f(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix},$$

then

$$\frac{\partial \mathbf{f}}{\partial x_j}(\mathbf{x}_0) = \begin{bmatrix} \frac{\partial f_1}{\partial x_j}(\mathbf{x}_0) \\ \vdots \\ \frac{\partial f_m}{\partial x_j}(\mathbf{x}_0) \end{bmatrix},$$

and the matrix L has the form

$$D\mathbf{f}(\mathbf{x}_0) = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1}(\mathbf{x}_0) & \cdots & \frac{\partial \mathbf{f}}{\partial x_n}(\mathbf{x}_0) \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}_0) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}_0) \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(\mathbf{x}_0) & \cdots & \frac{\partial f_m}{\partial x_n}(\mathbf{x}_0) \end{bmatrix} = L$$

The matrix L is called the *Jacobian matrix*, or *derivative matrix*, of f at x_0 , and is denoted $Df(x_0)$. For convenience, we often refer to $Df(x_0)$ simply as the derivative of f at x_0 . We summarize the foregoing discussion in the following theorem.

Theorem 5.5 If a function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is differentiable at x_0 , then the derivative of f at x_0 is determined uniquely and is represented by the $m \times n$ derivative matrix $Df(x_0)$. The best affine approximation to f near x_0 is then given by

$$\mathcal{A}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_0) + D\mathbf{f}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$$

in the sense that

$$f(x) = A(x) + r(x)$$

$$\frac{\|r(x)\|}{\|x - x_0\|} = 0$$

and $\lim_{\mathbf{x} \rightarrow \mathbf{x}_0} \|r(\mathbf{x})\|/\|\mathbf{x} - \mathbf{x}_0\| = 0$. The columns of the derivative matrix $D\mathbf{f}(\mathbf{x}_0)$ are vector partial derivatives. The vector

$$\frac{\partial \mathbf{f}}{\partial x_j}(\mathbf{x}_0)$$

is a tangent vector at \mathbf{x}_0 to the curve \mathbf{f} obtained by varying only the j th coordinate of \mathbf{x} . \square

If $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is differentiable, then the function ∇f defined by

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathbf{x}) \end{bmatrix} = Df(\mathbf{x})^\top$$

is called the *gradient* of f . The gradient is a function from \mathbb{R}^n to \mathbb{R}^n , and can be pictured as a *vector field*, by drawing the arrow representing $\nabla f(\mathbf{x})$ so that its tail starts at \mathbf{x} .

Given $f : \mathbb{R}^n \rightarrow \mathbb{R}$, if ∇f is differentiable, we say that f is *twice differentiable*, and we write the derivative of ∇f as

$$D^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

(The notation $\frac{\partial^2 f}{\partial x_i \partial x_j}$ represents taking the partial derivative of f with respect to x_j first, then with respect to x_i .) The matrix $D^2 f(\mathbf{x})$ is called the *Hessian matrix* of f at \mathbf{x} , and is often also denoted $F(\mathbf{x})$.

A function $\mathbf{f} : \Omega \rightarrow \mathbb{R}^m$, $\Omega \subset \mathbb{R}^n$, is said to be *continuously differentiable* on Ω if it is differentiable (on Ω), and $D\mathbf{f} : \Omega \rightarrow \mathbb{R}^{m \times n}$ is continuous; that is, the components of \mathbf{f} have continuous partial derivatives. In this case, we write $\mathbf{f} \in C^1$. If the components of \mathbf{f} have continuous partial derivatives of order p , then we write $\mathbf{f} \in C^p$.

Note that the Hessian matrix of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ at \mathbf{x} is symmetric if f is twice continuously differentiable at \mathbf{x} . This is a well-known result from calculus called *Clairaut's theorem* or *Schwarz's theorem*. However, if the second partial derivatives of f are not continuous, then there is no guarantee that the Hessian is symmetric, as shown in the following well-known example.

Example 5.1 Consider the function

$$f(\mathbf{x}) = \begin{cases} x_1 x_2 (x_1^2 - x_2^2) / (x_1^2 + x_2^2) & \text{if } \mathbf{x} \neq \mathbf{0} \\ 0 & \text{if } \mathbf{x} = \mathbf{0}. \end{cases}$$

$$\frac{x_1 x_2 (x_1^2 - x_2^2)}{x_1^2 + x_2^2}$$

$x_1 = 0$ $x_2 = 0$

?

Let us compute its Hessian at the point $\mathbf{0} = [0, 0]^\top$. We have

$$\mathbf{F} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_2 \partial x_1} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix}.$$

We now proceed with computing the components of the Hessian and evaluating them at the point $[0, 0]^\top$ one by one. We start with

$$\frac{\partial^2 f}{\partial x_1^2} = \frac{\partial}{\partial x_1} \left(\frac{\partial f}{\partial x_1} \right),$$

where

$$\begin{aligned} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ = \begin{cases} x_2(x_1^4 - x_2^4 + 4x_1^2x_2^2)/(x_1^2 + x_2^2)^2 & \text{if } \mathbf{x} \neq \mathbf{0} \\ 0 & \text{if } \mathbf{x} = \mathbf{0}. \end{cases} \end{aligned}$$

Note that

$$\frac{\partial f}{\partial x_1}([x_1, 0]^\top) = 0.$$

Hence,

$$\frac{\partial^2 f}{\partial x_1^2}(\mathbf{0}) = 0.$$

Also,

$$\frac{\partial f}{\partial x_1}([0, x_2]^\top) = -x_2.$$

Hence, the mixed partial is

$$\frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{0}) = -1.$$

We next compute

$$\frac{\partial^2 f}{\partial x_2^2} = \frac{\partial}{\partial x_2} \left(\frac{\partial f}{\partial x_2} \right),$$

where

$$\begin{aligned} \frac{\partial f}{\partial x_2}(x_1, x_2) \\ = \begin{cases} x_1(x_1^4 - x_2^4 - 4x_1^2x_2^2)/(x_1^2 + x_2^2)^2 & \text{if } \mathbf{x} \neq \mathbf{0} \\ 0 & \text{if } \mathbf{x} = \mathbf{0}. \end{cases} \end{aligned}$$

Note that

$$\frac{\partial f}{\partial x_2}([0, x_2]^\top) = 0.$$

Hence,

$$\frac{\partial^2 f}{\partial x_2^2}(\mathbf{0}) = 0.$$

Also,

$$\frac{\partial f}{\partial x_2}([x_1, 0]^\top) = x_1.$$

Hence, the mixed partial is

$$\frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{0}) = 1.$$

Therefore, the Hessian evaluated at the point $\mathbf{0}$ is

$$\mathbf{F}(\mathbf{0}) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

which is not symmetric. ■

5.4 Differentiation Rules

We now introduce the *chain rule* for differentiating the composition $g(\mathbf{f}(t))$, of a function $\mathbf{f} : \mathbb{R} \rightarrow \mathbb{R}^n$ and a function $g : \mathbb{R}^n \rightarrow \mathbb{R}$.

Theorem 5.6 Let $g : \mathcal{D} \rightarrow \mathbb{R}$ be differentiable on an open set $\mathcal{D} \subset \mathbb{R}^n$, and let $\mathbf{f} : (a, b) \rightarrow \mathcal{D}$ be differentiable on (a, b) . Then, the composite function $h : (a, b) \rightarrow \mathbb{R}$ given by $h(t) = g(\mathbf{f}(t))$ is differentiable on (a, b) , and

$$h'(t) = Dg(\mathbf{f}(t))D\mathbf{f}(t) = \nabla g(\mathbf{f}(t))^\top \begin{bmatrix} f'_1(t) \\ \vdots \\ f'_n(t) \end{bmatrix}.$$

□

Proof. By definition,

$$h'(t) = \lim_{s \rightarrow t} \frac{h(s) - h(t)}{s - t} = \lim_{s \rightarrow t} \frac{g(\mathbf{f}(s)) - g(\mathbf{f}(t))}{s - t}$$

if the limit exists. By Theorem 5.5 we write

$$g(\mathbf{f}(s)) - g(\mathbf{f}(t)) = Dg(\mathbf{f}(t))(\mathbf{f}(s) - \mathbf{f}(t)) + r(s),$$

where $\lim_{s \rightarrow t} r(s)/(s - t) = 0$. Therefore,

$$\frac{h(s) - h(t)}{s - t} = Dg(\mathbf{f}(t)) \frac{\mathbf{f}(s) - \mathbf{f}(t)}{s - t} + \frac{r(s)}{s - t}.$$

Letting $s \rightarrow t$ yields

$$h'(t) = \lim_{s \rightarrow t} Dg(\mathbf{f}(t)) \frac{\mathbf{f}(s) - \mathbf{f}(t)}{s - t} + \frac{r(s)}{s - t} = Dg(\mathbf{f}(t))D\mathbf{f}(t).$$

■

Next, we present the *product rule*. Let $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be two differentiable functions. Define the function $h : \mathbb{R}^n \rightarrow \mathbb{R}$ by $h(\mathbf{x}) = \mathbf{f}(\mathbf{x})^\top \mathbf{g}(\mathbf{x})$. Then, h is also differentiable and

$$Dh(\mathbf{x}) = \mathbf{f}(\mathbf{x})^\top D\mathbf{g}(\mathbf{x}) + \mathbf{g}(\mathbf{x})^\top D\mathbf{f}(\mathbf{x}).$$

z1 p2

We end this section with a list of some useful formulas from multivariable calculus. In each case, we compute the derivative with respect to \mathbf{x} . Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a given matrix and $\mathbf{y} \in \mathbb{R}^m$ a given vector. Then,

$$\left. \begin{array}{l} D(\mathbf{y}^\top \mathbf{A}\mathbf{x}) = \mathbf{y}^\top \mathbf{A} \\ D(\mathbf{x}^\top \mathbf{A}\mathbf{x}) = \mathbf{x}^\top (\mathbf{A} + \mathbf{A}^\top) \quad \text{if } m = n. \end{array} \right\}$$

It follows from the first formula above that if $\mathbf{y} \in \mathbb{R}^n$, then

$$D(\mathbf{y}^\top \mathbf{x}) = \mathbf{y}^\top.$$

It follows from the second formula above that if \mathbf{Q} is a symmetric matrix, then

$$D(\mathbf{x}^\top \mathbf{Q}\mathbf{x}) = 2\mathbf{x}^\top \mathbf{Q}.$$

In particular,

$$D(\mathbf{x}^\top \mathbf{x}) = 2\mathbf{x}^\top.$$

5.5 Level Sets and Gradients

The *level set* of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ at level c is the set of points

$$S = \{\mathbf{x} : f(\mathbf{x}) = c\}.$$

For $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, we are usually interested in S when it is a curve. For $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, the sets S most often considered are surfaces.

Example 5.2 Consider the following real-valued function on \mathbb{R}^2 :

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2, \quad \mathbf{x} = [x_1, x_2]^\top.$$

The function above is called *Rosenbrock's function*. A plot of the function f is shown in Figure 5.2. The level sets of f at levels 0.7, 7, 70, 200, and 700 are depicted in Figure 5.3. These level sets have a particular shape resembling

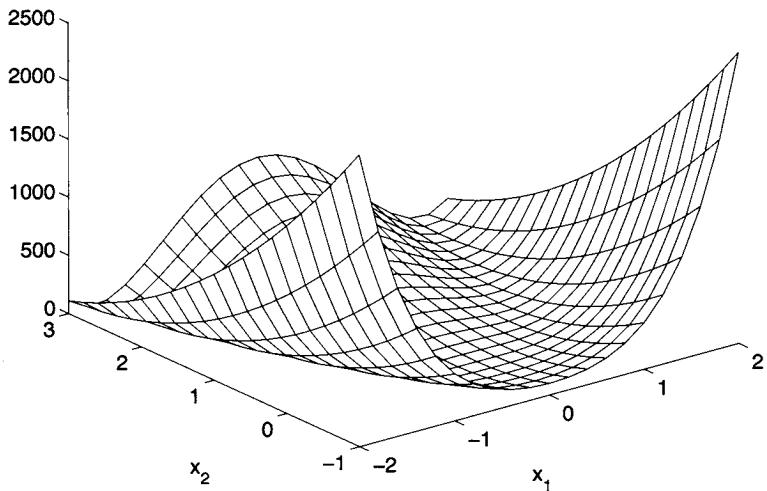


Figure 5.2 Graph of Rosenbrock's function.

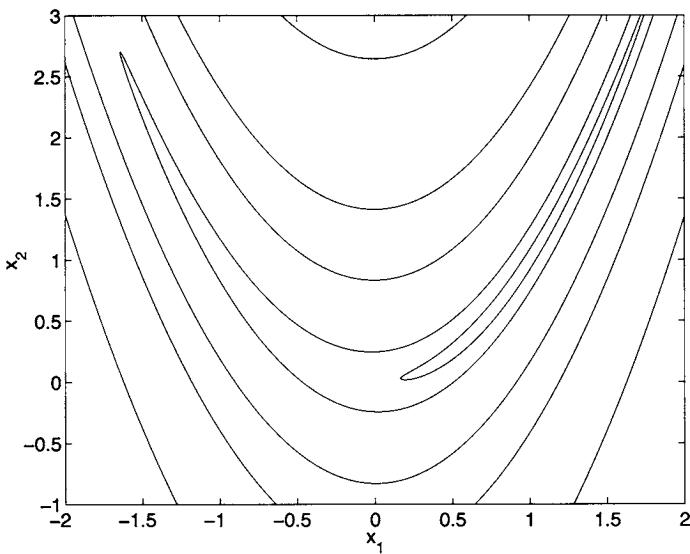


Figure 5.3 Level sets of Rosenbrock's (banana) function.

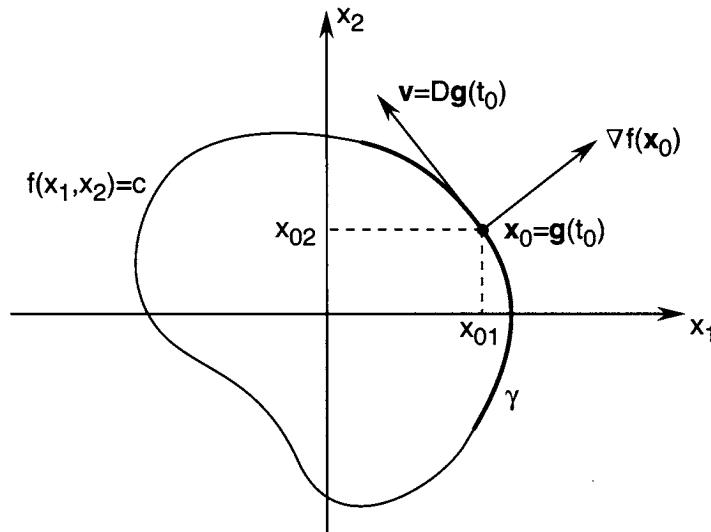


Figure 5.4 Orthogonality of the gradient to the level set.

bananas. For this reason, Rosenbrock's function is also called the *banana function*. ■

To say that a point \mathbf{x}_0 is on the level set S at level c means that $f(\mathbf{x}_0) = c$. Now suppose that there is a curve γ lying in S and parameterized by a continuously differentiable function $\mathbf{g} : \mathbb{R} \rightarrow \mathbb{R}^n$. Suppose also that $\mathbf{g}(t_0) = \mathbf{x}_0$ and $D\mathbf{g}(t_0) = \mathbf{v} \neq \mathbf{0}$, so that \mathbf{v} is a tangent vector to γ at \mathbf{x}_0 (see Figure 5.4). Applying the chain rule to the function $h(t) = f(\mathbf{g}(t))$ at t_0 gives

$$h'(t_0) = Df(\mathbf{g}(t_0))D\mathbf{g}(t_0) = Df(\mathbf{x}_0)\mathbf{v}.$$

But since γ lies on S , we have

$$h(t) = f(\mathbf{g}(t)) = c;$$

that is, h is constant. Thus, $h'(t_0) = 0$ and

$$Df(\mathbf{x}_0)\mathbf{v} = \nabla f(\mathbf{x}_0)^\top \mathbf{v} = 0.$$

Hence, we have proved, assuming f continuously differentiable, the following theorem (see Figure 5.4).

Theorem 5.7 *The vector $\nabla f(\mathbf{x}_0)$ is orthogonal to the tangent vector to an arbitrary smooth curve passing through \mathbf{x}_0 on the level set determined by $f(\mathbf{x}) = f(\mathbf{x}_0)$.* □

برهان اینجا $\nabla f(\mathbf{x}_0)$ عمود بر مجموعه سطحی $f(\mathbf{x}) = c$ است.

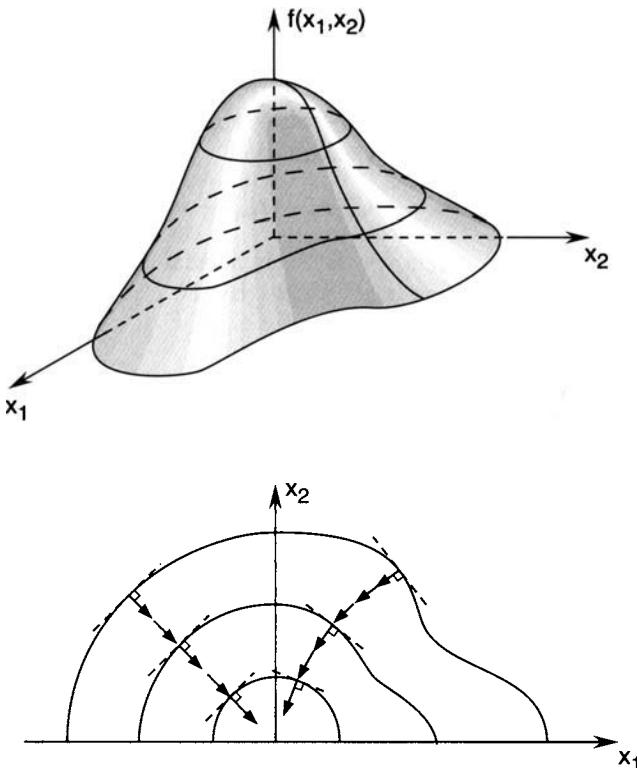


Figure 5.5 Illustration of a path of steepest ascent.

It is natural to say that $\nabla f(\mathbf{x}_0)$ is *orthogonal* or *normal* to the level set S corresponding to \mathbf{x}_0 , and it is also natural to take as the tangent plane (or line) to S at \mathbf{x}_0 the set of all points \mathbf{x} satisfying

$$\nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) = 0 \quad \text{if} \quad \nabla f(\mathbf{x}_0) \neq \mathbf{0}.$$

As we shall see later, $\nabla f(\mathbf{x}_0)$ is the direction of *maximum rate of increase* of f at \mathbf{x}_0 . Because $\nabla f(\mathbf{x}_0)$ is orthogonal to the level set through \mathbf{x}_0 determined by $f(\mathbf{x}) = f(\mathbf{x}_0)$, we deduce the following fact: The direction of maximum rate of increase of a real-valued differentiable function at a point is orthogonal to the level set of the function through that point.

Figure 5.5 illustrates the discussion above for the case $f : \mathbb{R}^2 \rightarrow \mathbb{R}$. The curve on the shaded surface in Figure 5.5 running from bottom to top has the property that its projection onto the (x_1, x_2) -plane is always orthogonal to the level curves and is called a *path of steepest ascent* because it always heads in the direction of maximum rate of increase for f .

The *graph* of $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the set $\{[\mathbf{x}^\top, f(\mathbf{x})]^\top : \mathbf{x} \in \mathbb{R}^n\} \subset \mathbb{R}^{n+1}$. The notion of the gradient of a function has an alternative useful interpretation

in terms of the tangent hyperplane to its graph. To proceed, let $\mathbf{x}_0 \in \mathbb{R}^n$ and $z_0 = f(\mathbf{x}_0)$. The point $[\mathbf{x}_0^\top, z_0]^\top \in \mathbb{R}^{n+1}$ is a point on the graph of f . If f is differentiable at ξ , then the graph admits a nonvertical tangent hyperplane at $\xi = [\mathbf{x}_0^\top, z_0]^\top$. The hyperplane through ξ is the set of all points $[x_1, \dots, x_n, z]^\top \in \mathbb{R}^{n+1}$ satisfying the equation

$$u_1(x_1 - x_{01}) + \dots + u_n(x_n - x_{0n}) + v(z - z_0) = 0,$$

where the vector $[u_1, \dots, u_n, v]^\top \in \mathbb{R}^{n+1}$ is normal to the hyperplane. Assuming that this hyperplane is nonvertical (that is, $v \neq 0$), let

$$d_i = -\frac{u_i}{v}.$$

Thus, we can rewrite the hyperplane equation above as

$$z = d_1(x_1 - x_{01}) + \dots + d_n(x_n - x_{0n}) + z_0.$$

We can think of the right side of the above equation as a function $z : \mathbb{R}^n \rightarrow \mathbb{R}$. Observe that for the hyperplane to be tangent to the graph of f , the functions f and z must have the same partial derivatives at the point \mathbf{x}_0 . Hence, if f is differentiable at \mathbf{x}_0 , its tangent hyperplane can be written in terms of its gradient, as given by the equation

$$z - z_0 = Df(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) = (\mathbf{x} - \mathbf{x}_0)^\top \nabla f(\mathbf{x}_0).$$

5.6 Taylor Series

The basis for many numerical methods and models for optimization is Taylor's formula, which is given by Taylor's theorem.

Theorem 5.8 Taylor's Theorem. Assume that a function $f : \mathbb{R} \rightarrow \mathbb{R}$ is m times continuously differentiable (i.e., $f \in C^m$) on an interval $[a, b]$. Denote $h = b - a$. Then,

$$f(b) = f(a) + \frac{h}{1!} f^{(1)}(a) + \frac{h^2}{2!} f^{(2)}(a) + \dots + \frac{h^{m-1}}{(m-1)!} f^{(m-1)}(a) + R_m,$$

(called Taylor's formula) where $f^{(i)}$ is the i th derivative of f , and

$$R_m = \frac{h^m (1-\theta)^{m-1}}{(m-1)!} f^{(m)}(a + \theta h) = \frac{h^m}{m!} f^{(m)}(a + \theta' h),$$

with $\theta, \theta' \in (0, 1)$. □

Proof. We have

$$R_m = f(b) - f(a) - \frac{h}{1!} f^{(1)}(a) - \frac{h^2}{2!} f^{(2)}(a) - \dots - \frac{h^{m-1}}{(m-1)!} f^{(m-1)}(a).$$

Denote by $g_m(x)$ an auxiliary function obtained from R_m by replacing a by x . Hence,

$$\begin{aligned} g_m(x) &= f(b) - f(x) - \frac{b-x}{1!} f^{(1)}(x) - \frac{(b-x)^2}{2!} f^{(2)}(x) \\ &\quad - \dots - \frac{(b-x)^{m-1}}{(m-1)!} f^{(m-1)}(x). \end{aligned}$$

Differentiating $g_m(x)$ yields

$$\begin{aligned} g_m^{(1)}(x) &= -f^{(1)}(x) + \left[f^{(1)}(x) - \frac{b-x}{1!} f^{(2)}(x) \right] \\ &\quad + \left[2 \frac{b-x}{2!} f^{(2)}(x) - \frac{(b-x)^2}{2!} f^{(3)}(x) \right] + \dots \\ &\quad + \left[(m-1) \frac{(b-x)^{m-2}}{(m-1)!} f^{(m-1)}(x) - \frac{(b-x)^{m-1}}{(m-1)!} f^{(m)}(x) \right] \\ &= -\frac{(b-x)^{m-1}}{(m-1)!} f^{(m)}(x). \end{aligned}$$

Observe that $g_m(b) = 0$ and $g_m(a) = R_m$. Applying the mean-value theorem yields

$$\frac{g_m(b) - g_m(a)}{b-a} = g_m^{(1)}(a + \theta h),$$

where $\theta \in (0, 1)$. The equation above is equivalent to

$$-\frac{R_m}{h} = -\frac{(b-a-\theta h)^{m-1}}{(m-1)!} f^{(m)}(a+\theta h) = -\frac{h^{m-1}(1-\theta)^{m-1}}{(m-1)!} f^{(m)}(a+\theta h).$$

Hence,

$$R_m = \frac{h^m(1-\theta)^{m-1}}{(m-1)!} f^{(m)}(a+\theta h).$$

To derive the formula

$$R_m = \frac{h^m}{m!} f^{(m)}(a+\theta' h),$$

see, e.g., [81] or [83]. ■

An important property of Taylor's theorem arises from the form of the remainder R_m . To discuss this property further, we introduce the *order symbols*, O and o .

Let g be a real-valued function defined in some neighborhood of $\mathbf{0} \in \mathbb{R}^n$, with $g(\mathbf{x}) \neq 0$ if $\mathbf{x} \neq \mathbf{0}$. Let $\mathbf{f} : \Omega \rightarrow \mathbb{R}^m$ be defined in a domain $\Omega \subset \mathbb{R}^n$ that includes $\mathbf{0}$. Then, we write

1. $\mathbf{f}(\mathbf{x}) = O(g(\mathbf{x}))$ to mean that the quotient $\|\mathbf{f}(\mathbf{x})\|/|g(\mathbf{x})|$ is bounded near 0; that is, there exist numbers $K > 0$ and $\delta > 0$ such that if $\|\mathbf{x}\| < \delta$, $\mathbf{x} \in \Omega$, then $\|\mathbf{f}(\mathbf{x})\|/|g(\mathbf{x})| \leq K$.

2. $\mathbf{f}(\mathbf{x}) = o(g(\mathbf{x}))$ to mean that

$$\lim_{\mathbf{x} \rightarrow 0, \mathbf{x} \in \Omega} \frac{\|\mathbf{f}(\mathbf{x})\|}{|g(\mathbf{x})|} = 0.$$

The symbol $O(g(\mathbf{x}))$ [read “big-oh of $g(\mathbf{x})$ ”] is used to represent a function that is bounded by a scaled version of g in a neighborhood of $\mathbf{0}$. Examples of such a function are:

- $x = O(x)$.
- $\begin{bmatrix} x^3 \\ 2x^2 + 3x^4 \end{bmatrix} = O(x^2)$.
- $\cos x = O(1)$.
- $\sin x = O(x)$.

On the other hand, $o(g(\mathbf{x}))$ [read “little-oh of $g(\mathbf{x})$ ”] represents a function that goes to zero “faster” than $g(\mathbf{x})$ in the sense that $\lim_{\mathbf{x} \rightarrow \mathbf{0}} \|o(g(\mathbf{x}))\|/|g(\mathbf{x})| = 0$. Examples of such functions are:

- $x^2 = o(x)$.
- $\begin{bmatrix} x^3 \\ 2x^2 + 3x^4 \end{bmatrix} = o(x)$.
- $x^3 = o(x^2)$.
- $x = o(1)$.

Note that if $\mathbf{f}(\mathbf{x}) = o(g(\mathbf{x}))$, then $\mathbf{f}(\mathbf{x}) = O(g(\mathbf{x}))$ (but the converse is not necessarily true). Also, if $\mathbf{f}(\mathbf{x}) = O(\|\mathbf{x}\|^p)$, then $\mathbf{f}(\mathbf{x}) = o(\|\mathbf{x}\|^{p-\varepsilon})$ for any $\varepsilon > 0$.

Suppose that $f \in \mathcal{C}^m$. Recall that the remainder term in Taylor’s theorem has the form

$$R_m = \frac{h^m}{m!} f^{(m)}(a + \theta h),$$

where $\theta \in (0, 1)$. Substituting this into Taylor’s formula, we get

$$f(b) = f(a) + \frac{h}{1!} f^{(1)}(a) + \frac{h^2}{2!} f^{(2)}(a) + \cdots + \frac{h^{m-1}}{(m-1)!} f^{(m-1)}(a) + \frac{h^m}{m!} f^{(m)}(a + \theta h).$$

By the continuity of $f^{(m)}$, we have $f^{(m)}(a + \theta h) \rightarrow f^{(m)}(a)$ as $h \rightarrow 0$; that is, $f^{(m)}(a + \theta h) = f^{(m)}(a) + o(1)$. Therefore,

$$\frac{h^m}{m!} f^{(m)}(a + \theta h) = \frac{h^m}{m!} f^{(m)}(a) + o(h^m),$$

since $h^m o(1) = o(h^m)$. We may then write Taylor's formula as

$$f(b) = f(a) + \frac{h}{1!} f^{(1)}(a) + \frac{h^2}{2!} f^{(2)}(a) + \cdots + \frac{h^m}{m!} f^{(m)}(a) + o(h^m).$$

If, in addition, we assume that $f \in \mathcal{C}^{m+1}$, we may replace the term $o(h^m)$ above by $O(h^{m+1})$. To see this, we first write Taylor's formula with R_{m+1} :

$$f(b) = f(a) + \frac{h}{1!} f^{(1)}(a) + \frac{h^2}{2!} f^{(2)}(a) + \cdots + \frac{h^m}{m!} f^{(m)}(a) + R_{m+1},$$

where

$$R_{m+1} = \frac{h^{m+1}}{(m+1)!} f^{(m+1)}(a + \theta' h),$$

with $\theta' \in (0, 1)$. Because $f^{(m+1)}$ is bounded on $[a, b]$ (by Theorem 4.2),

$$R_{m+1} = O(h^{m+1}).$$

Therefore, if $f \in \mathcal{C}^{m+1}$, we may write Taylor's formula as

$$f(b) = f(a) + \frac{h}{1!} f^{(1)}(a) + \frac{h^2}{2!} f^{(2)}(a) + \cdots + \frac{h^m}{m!} f^{(m)}(a) + O(h^{m+1}).$$

We now turn to the Taylor series expansion of a real-valued function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ about the point $\mathbf{x}_0 \in \mathbb{R}^n$. Suppose that $f \in \mathcal{C}^2$. Let \mathbf{x} and \mathbf{x}_0 be points in \mathbb{R}^n , and let $\mathbf{z}(\alpha) = \mathbf{x}_0 + \alpha(\mathbf{x} - \mathbf{x}_0)/\|\mathbf{x} - \mathbf{x}_0\|$. Define $\phi : \mathbb{R} \rightarrow \mathbb{R}$ by

$$\phi(\alpha) = f(\mathbf{z}(\alpha)) = f(\mathbf{x}_0 + \alpha(\mathbf{x} - \mathbf{x}_0)/\|\mathbf{x} - \mathbf{x}_0\|).$$

Using the chain rule, we obtain

$$\begin{aligned} \phi'(\alpha) &= \frac{d\phi}{d\alpha}(\alpha) \\ &= Df(\mathbf{z}(\alpha))D\mathbf{z}(\alpha) = Df(\mathbf{z}(\alpha))\frac{(\mathbf{x} - \mathbf{x}_0)}{\|\mathbf{x} - \mathbf{x}_0\|} \\ &= (\mathbf{x} - \mathbf{x}_0)^\top Df(\mathbf{z}(\alpha))^\top / \|\mathbf{x} - \mathbf{x}_0\| \end{aligned}$$

and

$$\begin{aligned}
 \phi''(\alpha) &= \frac{d^2\phi}{d\alpha^2}(\alpha) \\
 &= \frac{d}{d\alpha} \left(\frac{d\phi}{d\alpha} \right)(\alpha) \\
 &= \frac{(\mathbf{x} - \mathbf{x}_0)^\top}{\|\mathbf{x} - \mathbf{x}_0\|} \frac{d}{d\alpha} Df(\mathbf{z}(\alpha))^\top \\
 &= \frac{(\mathbf{x} - \mathbf{x}_0)^\top}{\|\mathbf{x} - \mathbf{x}_0\|} D(Df)(\mathbf{z}(\alpha))^\top \frac{d\mathbf{z}}{d\alpha}(\alpha) \\
 &= \frac{1}{\|\mathbf{x} - \mathbf{x}_0\|^2} (\mathbf{x} - \mathbf{x}_0)^\top D^2 f(\mathbf{z}(\alpha))^\top (\mathbf{x} - \mathbf{x}_0) \\
 &= \frac{1}{\|\mathbf{x} - \mathbf{x}_0\|^2} (\mathbf{x} - \mathbf{x}_0)^\top D^2 f(\mathbf{z}(\alpha))(\mathbf{x} - \mathbf{x}_0),
 \end{aligned}$$

where we recall that

$$D^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_2 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \frac{\partial^2 f}{\partial x_2 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Observe that

$$\begin{aligned}
 f(\mathbf{x}) &= \phi(\|\mathbf{x} - \mathbf{x}_0\|) \\
 &= \phi(0) + \frac{\|\mathbf{x} - \mathbf{x}_0\|}{1!} \phi'(0) + \frac{\|\mathbf{x} - \mathbf{x}_0\|^2}{2!} \phi''(0) + o(\|\mathbf{x} - \mathbf{x}_0\|^2).
 \end{aligned}$$

Hence,

$$\begin{aligned}
 f(\mathbf{x}) &= f(\mathbf{x}_0) + \frac{1}{1!} Df(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \\
 &\quad + \frac{1}{2!} (\mathbf{x} - \mathbf{x}_0)^\top D^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + o(\|\mathbf{x} - \mathbf{x}_0\|^2).
 \end{aligned}$$

If we assume that $f \in \mathcal{C}^3$, we may use the formula for the remainder term R_3 to conclude that

$$\begin{aligned}
 f(\mathbf{x}) &= f(\mathbf{x}_0) + \frac{1}{1!} Df(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \\
 &\quad + \frac{1}{2!} (\mathbf{x} - \mathbf{x}_0)^\top D^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) + O(\|\mathbf{x} - \mathbf{x}_0\|^3).
 \end{aligned}$$

We end with a statement of the *mean value theorem*, which is closely related to Taylor's theorem.

Theorem 5.9 If a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is differentiable on an open set $\Omega \subset \mathbb{R}^n$, then for any pair of points $\mathbf{x}, \mathbf{y} \in \Omega$, there exists a matrix \mathbf{M} such that

$$\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y}) = \mathbf{M}(\mathbf{x} - \mathbf{y}).$$

□

The mean value theorem follows from Taylor's theorem (for the case where $m = 1$) applied to each component of \mathbf{f} . It is easy to see that \mathbf{M} is a matrix whose rows are the rows of $D\mathbf{f}$ evaluated at points that lie on the line segment joining \mathbf{x} and \mathbf{y} (these points may differ from row to row).

For further reading in calculus, consult [13], [81], [83], [115], [120], [134]. A basic treatment of real analysis can be found in [2], [112], whereas a more advanced treatment is provided in [89], [111]. For stimulating reading on the “big-oh” notation, see [77, pp. 104–108].

EXERCISES

5.1 Show that a sufficient condition for $\lim_{k \rightarrow \infty} \mathbf{A}^k = \mathbf{O}$ is $\|\mathbf{A}\| < 1$.

5.2 Show that for any matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$,

$$\|\mathbf{A}\| \geq \max_{1 \leq i \leq n} |\lambda_i(\mathbf{A})|.$$

Hint: Use Exercise 5.1.

5.3 Consider the function

$$f(\mathbf{x}) = (\mathbf{a}^\top \mathbf{x})(\mathbf{b}^\top \mathbf{x}),$$

where \mathbf{a} , \mathbf{b} , and \mathbf{x} are n -dimensional vectors.

- a. Find $\nabla f(\mathbf{x})$.
- b. Find the Hessian $\mathbf{F}(\mathbf{x})$.

5.4 Define the functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ and $g : \mathbb{R} \rightarrow \mathbb{R}^2$ by $f(\mathbf{x}) = x_1^2/6 + x_2^2/4$, $\mathbf{g}(t) = [3t + 5, 2t - 6]^\top$. Let $F : \mathbb{R} \rightarrow \mathbb{R}$ be given by $F(t) = f(g(t))$. Evaluate $\frac{dF}{dt}(t)$ using the chain rule.

5.5 Consider $f(\mathbf{x}) = x_1x_2/2$, $\mathbf{g}(s, t) = [4s + 3t, 2s + t]^\top$. Evaluate $\frac{\partial}{\partial s} f(\mathbf{g}(s, t))$ and $\frac{\partial}{\partial t} f(\mathbf{g}(s, t))$ using the chain rule.

5.6 Let $\mathbf{x}(t) = [e^t + t^3, t^2, t + 1]^\top$, $t \in \mathbb{R}$, and $f(\mathbf{x}) = x_1^3x_2x_3^2 + x_1x_2 + x_3$, $\mathbf{x} = [x_1, x_2, x_3]^\top \in \mathbb{R}^3$. Find $\frac{d}{dt} f(\mathbf{x}(t))$ in terms of t .

5.7 Suppose that $\mathbf{f}(\mathbf{x}) = o(g(\mathbf{x}))$. Show that for any given $\varepsilon > 0$, there exists $\delta > 0$ such that if $\|\mathbf{x}\| < \delta$, then $\|\mathbf{f}(\mathbf{x})\| < \varepsilon|g(\mathbf{x})|$.

5.8 Use Exercise 5.7 to show that if functions $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}$ satisfy $f(\mathbf{x}) = -g(\mathbf{x}) + o(g(\mathbf{x}))$ and $g(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{0}$, then for all $\mathbf{x} \neq \mathbf{0}$ sufficiently small, we have $f(\mathbf{x}) < 0$.

5.9 Let

$$\begin{aligned} f_1(x_1, x_2) &= x_1^2 - x_2^2, \\ f_2(x_1, x_2) &= 2x_1x_2. \end{aligned}$$

Sketch the level sets associated with $f_1(x_1, x_2) = 12$ and $f_2(x_1, x_2) = 16$ on the same diagram. Indicate on the diagram the values of $\mathbf{x} = [x_1, x_2]^\top$ for which $\mathbf{f}(\mathbf{x}) = [f_1(x_1, x_2), f_2(x_1, x_2)]^\top = [12, 16]^\top$.

5.10 Write down the Taylor series expansion of the following functions about the given points \mathbf{x}_0 . Neglect terms of order three or higher.

- a. $f(\mathbf{x}) = x_1 e^{-x_2} + x_2 + 1$, $\mathbf{x}_0 = [1, 0]^\top$.
- b. $f(\mathbf{x}) = x_1^4 + 2x_1^2x_2^2 + x_2^4$, $\mathbf{x}_0 = [1, 1]^\top$.
- c. $f(\mathbf{x}) = e^{x_1-x_2} + e^{x_1+x_2} + x_1 + x_2 + 1$, $\mathbf{x}_0 = [1, 0]^\top$.

PART II

UNCONSTRAINED OPTIMIZATION

CHAPTER 6

BASICS OF SET-CONSTRAINED AND UNCONSTRAINED OPTIMIZATION

6.1 Introduction

In this chapter we consider the optimization problem

$$\begin{aligned} & \text{minimize} && f(\boldsymbol{x}) \\ & \text{subject to} && \boldsymbol{x} \in \Omega. \end{aligned}$$

The function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that we wish to minimize is a real-valued function called the *objective function* or *cost function*. The vector \boldsymbol{x} is an n -vector of independent variables: $\boldsymbol{x} = [x_1, x_2, \dots, x_n]^\top \in \mathbb{R}^n$. The variables x_1, \dots, x_n are often referred to as *decision variables*. The set Ω is a subset of \mathbb{R}^n called the *constraint set* or *feasible set*.

The optimization problem above can be viewed as a decision problem that involves finding the “best” vector \boldsymbol{x} of the decision variables over all possible vectors in Ω . By the “best” vector we mean the one that results in the smallest value of the objective function. This vector is called the *minimizer* of f over Ω . It is possible that there may be many minimizers. In this case, finding any of the minimizers will suffice.

There are also optimization problems that require maximization of the objective function, in which case we seek *maximizers*. Minimizers and maximizers are also called *extremizers*. Maximization problems, however, can be represented equivalently in the minimization form above because maximizing f is equivalent to minimizing $-f$. Therefore, we can confine our attention to minimization problems without loss of generality.

The problem above is a general form of a *constrained optimization problem*, because the decision variables are constrained to be in the constraint set Ω . If $\Omega = \mathbb{R}^n$, then we refer to the problem as an *unconstrained optimization problem*. In this chapter we discuss basic properties of the general optimization problem above, which includes the unconstrained case. In the remaining chapters of this part, we deal with iterative algorithms for solving unconstrained optimization problems.

The constraint " $\mathbf{x} \in \Omega$ " is called a *set constraint*. Often, the constraint set Ω takes the form $\Omega = \{\mathbf{x} : \mathbf{h}(\mathbf{x}) = \mathbf{0}, \mathbf{g}(\mathbf{x}) \leq \mathbf{0}\}$, where \mathbf{h} and \mathbf{g} are given functions. We refer to such constraints as *functional constraints*. The remainder of this chapter deals with general set constraints, including the special case where $\Omega = \mathbb{R}^n$. The case where $\Omega = \mathbb{R}^n$ is called the *unconstrained* case. In Parts III and IV we consider constrained optimization problems with functional constraints.

In considering the general optimization problem above, we distinguish between two kinds of minimizers, as specified by the following definitions.

Definition 6.1 Suppose that $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a real-valued function defined on some set $\Omega \subset \mathbb{R}^n$. A point $\mathbf{x}^* \in \Omega$ is a *local minimizer* of f over Ω if there exists $\varepsilon > 0$ such that $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ for all $\mathbf{x} \in \Omega \setminus \{\mathbf{x}^*\}$ and $\|\mathbf{x} - \mathbf{x}^*\| < \varepsilon$. A point $\mathbf{x}^* \in \Omega$ is a *global minimizer* of f over Ω if $f(\mathbf{x}) \geq f(\mathbf{x}^*)$ for all $\mathbf{x} \in \Omega \setminus \{\mathbf{x}^*\}$. ■

If in the definitions above we replace " \geq " with " $>$," then we have a *strict local minimizer* and a *strict global minimizer*, respectively. In Figure 6.1, we illustrate the definitions for $n = 1$.

If \mathbf{x}^* is a global minimizer of f over Ω , we write $f(\mathbf{x}^*) = \min_{\mathbf{x} \in \Omega} f(\mathbf{x})$ and $\mathbf{x}^* = \arg \min_{\mathbf{x} \in \Omega} f(\mathbf{x})$. If the minimization is unconstrained, we simply write $\mathbf{x}^* = \arg \min_{\mathbf{x}} f(\mathbf{x})$ or $\mathbf{x}^* = \arg \min f(\mathbf{x})$. In other words, given a real-valued function f , the notation $\arg \min f(\mathbf{x})$ denotes the *argument* that minimizes the function f (a point in the domain of f), assuming that such a point is unique (if there is more than one such point, we pick one arbitrarily). For example, if $f : \mathbb{R} \rightarrow \mathbb{R}$ is given by $f(x) = (x+1)^2 + 3$, then $\arg \min f(x) = -1$. If we write $\arg \min_{x \in \Omega}$, then we treat " $x \in \Omega$ " to be a constraint for the minimization. For example, for the function f above, $\arg \min_{x \geq 0} f(x) = 0$.

Strictly speaking, an optimization problem is solved only when a global minimizer is found. However, global minimizers are, in general, difficult to find. Therefore, in practice, we often have to be satisfied with finding local minimizers.

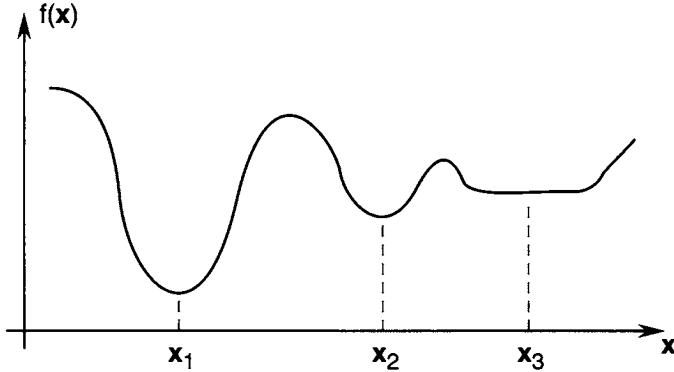


Figure 6.1 Examples of minimizers: \mathbf{x}_1 : strict global minimizer; \mathbf{x}_2 : strict local minimizer; \mathbf{x}_3 : local (not strict) minimizer.

6.2 Conditions for Local Minimizers

In this section we derive conditions for a point \mathbf{x}^* to be a local minimizer. We use derivatives of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. Recall that the first-order derivative of f , denoted Df , is

$$Df \triangleq \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right].$$

Note that the gradient ∇f is just the transpose of Df ; that is, $\nabla f = (Df)^\top$. The second derivative of $f : \mathbb{R}^n \rightarrow \mathbb{R}$ (also called the *Hessian* of f) is

$$\mathbf{F}(\mathbf{x}) \triangleq D^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1}(\mathbf{x}) \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(\mathbf{x}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\mathbf{x}) \end{bmatrix}.$$

Example 6.1 Let $f(x_1, x_2) = 5x_1 + 8x_2 + x_1x_2 - x_1^2 - 2x_2^2$. Then,

$$Df(\mathbf{x}) = (\nabla f(\mathbf{x}))^\top = \left[\frac{\partial f}{\partial x_1}(\mathbf{x}), \frac{\partial f}{\partial x_2}(\mathbf{x}) \right] = [5 + x_2 - 2x_1, 8 + x_1 - 4x_2]$$

and

$$\mathbf{F}(\mathbf{x}) = D^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{x}) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2^2}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1 & -4 \end{bmatrix}.$$

■

Given an optimization problem with constraint set Ω , a minimizer may lie either in the interior or on the boundary of Ω . To study the case where it lies on the boundary, we need the notion of *feasible directions*.

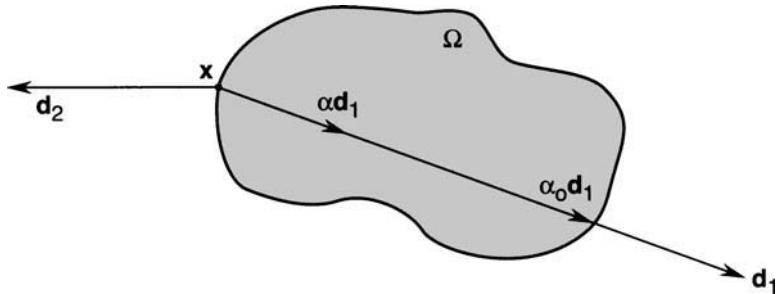


Figure 6.2 Two-dimensional illustration of feasible directions; \mathbf{d}_1 is a feasible direction, \mathbf{d}_2 is not a feasible direction.

Definition 6.2 A vector $\mathbf{d} \in \mathbb{R}^n$, $\mathbf{d} \neq \mathbf{0}$, is a *feasible direction* at $\mathbf{x} \in \Omega$ if there exists $\alpha_0 > 0$ such that $\mathbf{x} + \alpha\mathbf{d} \in \Omega$ for all $\alpha \in [0, \alpha_0]$. ■

Figure 6.2 illustrates the notion of feasible directions.

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a real-valued function and let \mathbf{d} be a feasible direction at $\mathbf{x} \in \Omega$. The *directional derivative of f in the direction \mathbf{d}* , denoted $\partial f / \partial \mathbf{d}$, is the real-valued function defined by

$$\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}) = \lim_{\alpha \rightarrow 0} \frac{f(\mathbf{x} + \alpha\mathbf{d}) - f(\mathbf{x})}{\alpha}.$$

If $\|\mathbf{d}\| = 1$, then $\partial f / \partial \mathbf{d}$ is the rate of increase of f at \mathbf{x} in the direction \mathbf{d} . To compute the directional derivative above, suppose that \mathbf{x} and \mathbf{d} are given. Then, $f(\mathbf{x} + \alpha\mathbf{d})$ is a function of α , and

$$\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}) = \frac{d}{d\alpha} f(\mathbf{x} + \alpha\mathbf{d}) \Big|_{\alpha=0}.$$

Applying the chain rule yields

$$\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}) = \frac{d}{d\alpha} f(\mathbf{x} + \alpha\mathbf{d}) \Big|_{\alpha=0} = \nabla f(\mathbf{x})^\top \mathbf{d} = \langle \nabla f(\mathbf{x}), \mathbf{d} \rangle = \mathbf{d}^\top \nabla f(\mathbf{x}).$$

In summary, if \mathbf{d} is a unit vector ($\|\mathbf{d}\| = 1$), then $\langle \nabla f(\mathbf{x}), \mathbf{d} \rangle$ is the rate of increase of f at the point \mathbf{x} in the direction \mathbf{d} .

Example 6.2 Define $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ by $f(\mathbf{x}) = x_1 x_2 x_3$, and let

$$\mathbf{d} = \left[\frac{1}{2}, \frac{1}{2}, \frac{1}{\sqrt{2}} \right]^\top.$$

The directional derivative of f in the direction \mathbf{d} is

$$\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}) = \nabla f(\mathbf{x})^\top \mathbf{d} = [x_2 x_3, x_1 x_3, x_1 x_2] \begin{bmatrix} 1/2 \\ 1/2 \\ 1/\sqrt{2} \end{bmatrix} = \frac{x_2 x_3 + x_1 x_3 + \sqrt{2} x_1 x_2}{2}.$$

Note that because $\|\mathbf{d}\| = 1$, the above is also the rate of increase of f at \mathbf{x} in the direction \mathbf{d} . ■

We are now ready to state and prove the following theorem.

Theorem 6.1 First-Order Necessary Condition (FONC). *Let Ω be a subset of \mathbb{R}^n and $f \in C^1$ a real-valued function on Ω . If \mathbf{x}^* is a local minimizer of f over Ω , then for any feasible direction \mathbf{d} at \mathbf{x}^* , we have*

$$\mathbf{d}^\top \nabla f(\mathbf{x}^*) \geq 0.$$

□

Proof. Define

$$\mathbf{x}(\alpha) = \mathbf{x}^* + \alpha\mathbf{d} \in \Omega.$$

Note that $\mathbf{x}(0) = \mathbf{x}^*$. Define the composite function

$$\phi(\alpha) = f(\mathbf{x}(\alpha)).$$

Then, by Taylor's theorem,

$$f(\mathbf{x}^* + \alpha\mathbf{d}) - f(\mathbf{x}^*) = \phi(\alpha) - \phi(0) = \phi'(0)\alpha + o(\alpha) = \alpha\mathbf{d}^\top \nabla f(\mathbf{x}(0)) + o(\alpha),$$

where $\alpha \geq 0$ [recall the definition of $o(\alpha)$ ("little-oh of α ") in Part I]. Thus, if $\phi(\alpha) \geq \phi(0)$, that is, $f(\mathbf{x}^* + \alpha\mathbf{d}) \geq f(\mathbf{x}^*)$ for sufficiently small values of $\alpha > 0$ (\mathbf{x}^* is a local minimizer), then we have to have $\mathbf{d}^\top \nabla f(\mathbf{x}^*) \geq 0$ (see Exercise 5.8). ■

Theorem 6.1 is illustrated in Figure 6.3.

An alternative way to express the FONC is

$$\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}^*) \geq 0$$

for all feasible directions \mathbf{d} . In other words, if \mathbf{x}^* is a local minimizer, then the rate of increase of f at \mathbf{x}^* in any feasible direction \mathbf{d} in Ω is nonnegative. Using directional derivatives, an alternative proof of Theorem 6.1 is as follows. Suppose that \mathbf{x}^* is a local minimizer. Then, for any feasible direction \mathbf{d} , there exists $\bar{\alpha} > 0$ such that for all $\alpha \in (0, \bar{\alpha})$,

$$f(\mathbf{x}^*) \leq f(\mathbf{x}^* + \alpha\mathbf{d}).$$

Hence, for all $\alpha \in (0, \bar{\alpha})$, we have

$$\frac{f(\mathbf{x}^* + \alpha\mathbf{d}) - f(\mathbf{x}^*)}{\alpha} \geq 0.$$

Taking the limit as $\alpha \rightarrow 0$, we conclude that

$$\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}^*) \geq 0.$$

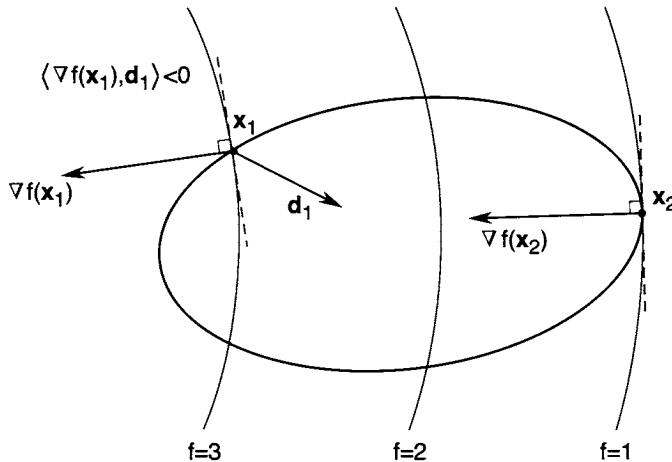


Figure 6.3 Illustration of the FONC for a constrained case; x_1 does not satisfy the FONC, whereas x_2 satisfies the FONC.

A special case of interest is when \mathbf{x}^* is an interior point of Ω (see Section 4.4). In this case, any direction is feasible, and we have the following result.

Corollary 6.1 Interior Case. Let Ω be a subset of \mathbb{R}^n and $f \in \mathcal{C}^1$ a real-valued function on Ω . If \mathbf{x}^* is a local minimizer of f over Ω and if \mathbf{x}^* is an interior point of Ω , then

$$\nabla f(\mathbf{x}^*) = \mathbf{0}.$$

□

Proof. Suppose that f has a local minimizer \mathbf{x}^* that is an interior point of Ω . Because \mathbf{x}^* is an interior point of Ω , the set of feasible directions at \mathbf{x}^* is the whole of \mathbb{R}^n . Thus, for any $\mathbf{d} \in \mathbb{R}^n$, $\mathbf{d}^\top \nabla f(\mathbf{x}^*) \geq 0$ and $-\mathbf{d}^\top \nabla f(\mathbf{x}^*) \geq 0$. Hence, $\mathbf{d}^\top \nabla f(\mathbf{x}^*) = 0$ for all $\mathbf{d} \in \mathbb{R}^n$, which implies that $\nabla f(\mathbf{x}^*) = \mathbf{0}$. ■

Example 6.3 Consider the problem

$$\begin{aligned} \text{minimize } & x_1^2 + 0.5x_2^2 + 3x_2 + 4.5 \\ \text{subject to } & x_1, x_2 \geq 0. \end{aligned}$$

- a. Is the first-order necessary condition (FONC) for a local minimizer satisfied at $\mathbf{x} = [1, 3]^\top$?
- b. Is the FONC for a local minimizer satisfied at $\mathbf{x} = [0, 3]^\top$?
- c. Is the FONC for a local minimizer satisfied at $\mathbf{x} = [1, 0]^\top$?

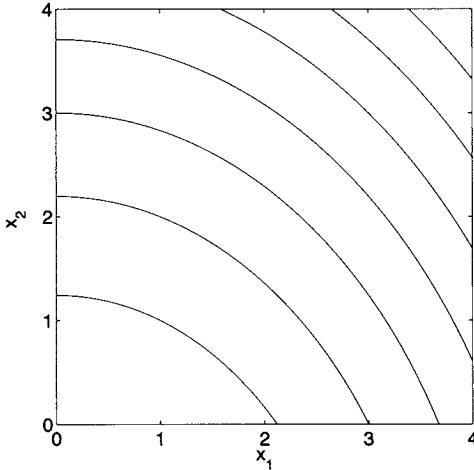


Figure 6.4 Level sets of the function in Example 6.3.

- d. Is the FONC for a local minimizer satisfied at $\mathbf{x} = [0, 0]^\top$?

Solution: First, let $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ be defined by $f(\mathbf{x}) = x_1^2 + 0.5x_2^2 + 3x_2 + 4.5$, where $\mathbf{x} = [x_1, x_2]^\top$. A plot of the level sets of f is shown in Figure 6.4.

- a. At $\mathbf{x} = [1, 3]^\top$, we have $\nabla f(\mathbf{x}) = [2x_1, x_2 + 3]^\top = [2, 6]^\top$. The point $\mathbf{x} = [1, 3]^\top$ is an interior point of $\Omega = \{\mathbf{x} : x_1 \geq 0, x_2 \geq 0\}$. Hence, the FONC requires that $\nabla f(\mathbf{x}) = \mathbf{0}$. The point $\mathbf{x} = [1, 3]^\top$ does not satisfy the FONC for a local minimizer.
- b. At $\mathbf{x} = [0, 3]^\top$, we have $\nabla f(\mathbf{x}) = [0, 6]^\top$, and hence $\mathbf{d}^\top \nabla f(\mathbf{x}) = 6d_2$, where $\mathbf{d} = [d_1, d_2]^\top$. For \mathbf{d} to be feasible at \mathbf{x} , we need $d_1 \geq 0$, and d_2 can take an arbitrary value in \mathbb{R} . The point $\mathbf{x} = [0, 3]^\top$ does not satisfy the FONC for a minimizer because d_2 is allowed to be less than zero. For example, $\mathbf{d} = [1, -1]^\top$ is a feasible direction, but $\mathbf{d}^\top \nabla f(\mathbf{x}) = -6 < 0$.
- c. At $\mathbf{x} = [1, 0]^\top$, we have $\nabla f(\mathbf{x}) = [2, 3]^\top$, and hence $\mathbf{d}^\top \nabla f(\mathbf{x}) = 2d_1 + 3d_2$. For \mathbf{d} to be feasible, we need $d_2 \geq 0$, and d_1 can take an arbitrary value in \mathbb{R} . For example, $\mathbf{d} = [-5, 1]^\top$ is a feasible direction. But $\mathbf{d}^\top \nabla f(\mathbf{x}) = -7 < 0$. Thus, $\mathbf{x} = [1, 0]^\top$ does not satisfy the FONC for a local minimizer.
- d. At $\mathbf{x} = [0, 0]^\top$, we have $\nabla f(\mathbf{x}) = [0, 3]^\top$, and hence $\mathbf{d}^\top \nabla f(\mathbf{x}) = 3d_2$. For \mathbf{d} to be feasible, we need $d_2 \geq 0$ and $d_1 \geq 0$. Hence, $\mathbf{x} = [0, 0]^\top$ satisfies the FONC for a local minimizer. ■

Example 6.4 Figure 6.5 shows a simplified model of a cellular wireless system (the distances shown have been scaled down to make the calculations

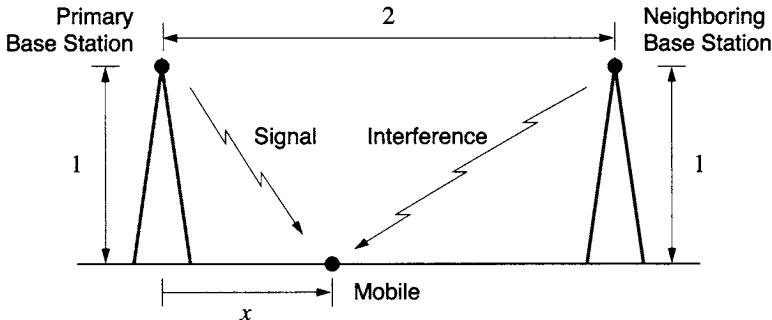


Figure 6.5 Simplified cellular wireless system in Example 6.4.

simpler). A mobile user (also called a *mobile*) is located at position x (see Figure 6.5).

There are two base station antennas, one for the primary base station and another for the neighboring base station. Both antennas are transmitting signals to the mobile user, at equal power. However, the power of the received signal as measured by the mobile is the reciprocal of the squared distance from the associated antenna (primary or neighboring base station). We are interested in finding the position of the mobile that maximizes the *signal-to-interference ratio*, which is the ratio of the signal power received from the primary base station to the signal power received from the neighboring base station.

We use the FONC to solve this problem. The squared distance from the mobile to the primary antenna is $1 + x^2$, while the squared distance from the mobile to the neighboring antenna is $1 + (2 - x)^2$. Therefore, the signal-to-interference ratio is

$$f(x) = \frac{1 + (2 - x)^2}{1 + x^2}.$$

We have

$$\begin{aligned} f'(x) &= \frac{-2(2-x)(1+x^2) - 2x(1+(2-x)^2)}{(1+x^2)^2} \\ &= \frac{4(x^2 - 2x - 1)}{(1+x^2)^2}. \end{aligned}$$

By the FONC, at the optimal position x^* we have $f'(x^*) = 0$. Hence, either $x^* = 1 - \sqrt{2}$ or $x^* = 1 + \sqrt{2}$. Evaluating the objective function at these two candidate points, it is easy to see that $x^* = 1 - \sqrt{2}$ is the optimal position. ■

The next example illustrates that in some problems the FONC is not helpful for eliminating candidate local minimizers. However, in such cases, there may be a recasting of the problem into an equivalent form that makes the FONC useful.

Example 6.5 Consider the set-constrained problem

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}) \\ & \text{subject to} && \mathbf{x} \in \Omega, \end{aligned}$$

where $\Omega = \{[\mathbf{x}_1, \mathbf{x}_2]^\top : \mathbf{x}_1^2 + \mathbf{x}_2^2 = 1\}$.

- a. Consider a point $\mathbf{x}^* \in \Omega$. Specify all feasible directions at \mathbf{x}^* .
- b. Which points in Ω satisfy the FONC for this set-constrained problem?
- c. Based on part b, is the FONC for this set-constrained problem useful for eliminating local-minimizer candidates?
- d. Suppose that we use polar coordinates to parameterize points $\mathbf{x} \in \Omega$ in terms of a single parameter θ :

$$x_1 = \cos \theta \quad x_2 = \sin \theta.$$

Now use the FONC for unconstrained problems (with respect to θ) to derive a necessary condition of this sort: If $\mathbf{x}^* \in \Omega$ is a local minimizer, then $\mathbf{d}^\top \nabla f(\mathbf{x}^*) = 0$ for all \mathbf{d} satisfying a “certain condition.” Specify what this certain condition is.

Solution:

- a. There are no feasible directions at any \mathbf{x}^* .
- b. Because of part a, *all* points in Ω satisfy the FONC for this set-constrained problem.
- c. No, the FONC for this set-constrained problem is not useful for eliminating local-minimizer candidates.
- d. Write $h(\theta) = f(g(\theta))$, where $g : \mathbb{R} \rightarrow \mathbb{R}^2$ is given by the equations relating θ to $\mathbf{x} = [x_1, x_2]^\top$. Note that $Dg(\theta) = [-\sin \theta, \cos \theta]^\top$. Hence, by the chain rule,

$$h'(\theta) = Df(g(\theta))Dg(\theta) = Dg(\theta)^\top \nabla f(g(\theta)).$$

Notice that $Dg(\theta)$ is tangent to Ω at $\mathbf{x} = g(\theta)$. Alternatively, we could say that $Dg(\theta)$ is orthogonal to $\mathbf{x} = g(\theta)$.

Suppose that $\mathbf{x}^* \in \Omega$ is a local minimizer. Write $\mathbf{x}^* = g(\theta^*)$. Then θ^* is an unconstrained minimizer of h . By the FONC for unconstrained problems, $h'(\theta^*) = 0$, which implies that $\mathbf{d}^\top \nabla f(\mathbf{x}^*) = 0$ for all \mathbf{d} tangent to Ω at \mathbf{x}^* (or, alternatively, for all \mathbf{d} orthogonal to \mathbf{x}^*). ■

We now derive a second-order necessary condition that is satisfied by a local minimizer.

Theorem 6.2 Second-Order Necessary Condition (SONC). Let $\Omega \subset \mathbb{R}^n$, $f \in \mathcal{C}^2$ a function on Ω , \mathbf{x}^* a local minimizer of f over Ω , and \mathbf{d} a feasible direction at \mathbf{x}^* . If $\mathbf{d}^\top \nabla f(\mathbf{x}^*) = 0$, then

$$\mathbf{d}^\top \mathbf{F}(\mathbf{x}^*) \mathbf{d} \geq 0,$$

where \mathbf{F} is the Hessian of f . □

Proof. We prove the result by contradiction. Suppose that there is a feasible direction \mathbf{d} at \mathbf{x}^* such that $\mathbf{d}^\top \nabla f(\mathbf{x}^*) = 0$ and $\mathbf{d}^\top \mathbf{F}(\mathbf{x}^*) \mathbf{d} < 0$. Let $\mathbf{x}(\alpha) = \mathbf{x}^* + \alpha \mathbf{d}$ and define the composite function $\phi(\alpha) = f(\mathbf{x}^* + \alpha \mathbf{d}) = f(\mathbf{x}(\alpha))$. Then, by Taylor's theorem,

$$\phi(\alpha) = \phi(0) + \phi''(0) \frac{\alpha^2}{2} + o(\alpha^2),$$

where by assumption, $\phi'(0) = \mathbf{d}^\top \nabla f(\mathbf{x}^*) = 0$ and $\phi''(0) = \mathbf{d}^\top \mathbf{F}(\mathbf{x}^*) \mathbf{d} < 0$. For sufficiently small α ,

$$\phi(\alpha) - \phi(0) = \phi''(0) \frac{\alpha^2}{2} + o(\alpha^2) < 0,$$

that is,

$$f(\mathbf{x}^* + \alpha \mathbf{d}) < f(\mathbf{x}^*),$$

which contradicts the assumption that \mathbf{x}^* is a local minimizer. Thus,

$$\phi''(0) = \mathbf{d}^\top \mathbf{F}(\mathbf{x}^*) \mathbf{d} \geq 0.$$
■

Corollary 6.2 Interior Case. Let \mathbf{x}^* be an interior point of $\Omega \subset \mathbb{R}^n$. If \mathbf{x}^* is a local minimizer of $f : \Omega \rightarrow \mathbb{R}$, $f \in \mathcal{C}^2$, then

$$\nabla f(\mathbf{x}^*) = \mathbf{0},$$

and $\mathbf{F}(\mathbf{x}^*)$ is positive semidefinite ($\mathbf{F}(\mathbf{x}^*) \geq 0$); that is, for all $\mathbf{d} \in \mathbb{R}^n$,

$$\mathbf{d}^\top \mathbf{F}(\mathbf{x}^*) \mathbf{d} \geq 0.$$
□

Proof. If \mathbf{x}^* is an interior point, then all directions are feasible. The result then follows from Corollary 6.1 and Theorem 6.2. ■

In the examples below, we show that the necessary conditions are *not* sufficient.

Example 6.6 Consider a function of one variable $f(x) = x^3$, $f : \mathbb{R} \rightarrow \mathbb{R}$. Because $f'(0) = 0$, and $f''(0) = 0$, the point $x = 0$ satisfies both the FONC and SONC. However, $x = 0$ is not a minimizer (see Figure 6.6). ■

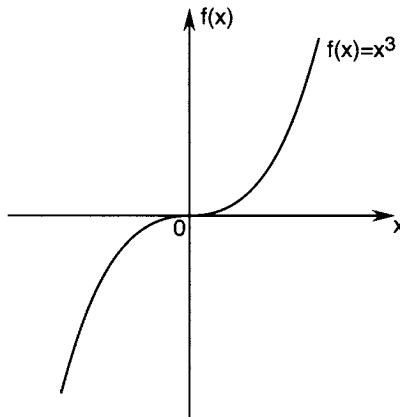


Figure 6.6 The point 0 satisfies the FONC and SONC but is not a minimizer.

Example 6.7 Consider a function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, where $f(\mathbf{x}) = x_1^2 - x_2^2$. The FONC requires that $\nabla f(\mathbf{x}) = [2x_1, -2x_2]^\top = \mathbf{0}$. Thus, $\mathbf{x} = [0, 0]^\top$ satisfies the FONC. The Hessian matrix of f is

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}.$$

The Hessian matrix is indefinite; that is, for some $\mathbf{d}_1 \in \mathbb{R}^2$ we have $\mathbf{d}_1^\top \mathbf{F} \mathbf{d}_1 > 0$ (e.g., $\mathbf{d}_1 = [1, 0]^\top$) and for some \mathbf{d}_2 we have $\mathbf{d}_2^\top \mathbf{F} \mathbf{d}_2 < 0$ (e.g., $\mathbf{d}_2 = [0, 1]^\top$). Thus, $\mathbf{x} = [0, 0]^\top$ does not satisfy the SONC, and hence it is not a minimizer. The graph of $f(\mathbf{x}) = x_1^2 - x_2^2$ is shown in Figure 6.7. ■

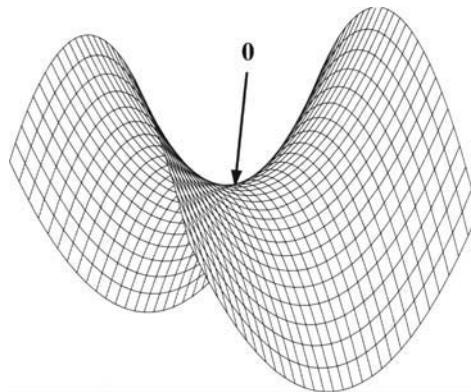


Figure 6.7 Graph of $f(\mathbf{x}) = x_1^2 - x_2^2$. The point $\mathbf{0}$ satisfies the FONC but not SONC; this point is not a minimizer.

We now derive sufficient conditions that imply that \mathbf{x}^* is a local minimizer.

Theorem 6.3 Second-Order Sufficient Condition (SOSC), Interior Case. Let $f \in \mathcal{C}^2$ be defined on a region in which \mathbf{x}^* is an interior point. Suppose that

1. $\nabla f(\mathbf{x}^*) = \mathbf{0}$.
2. $\mathbf{F}(\mathbf{x}^*) > 0$.

Then, \mathbf{x}^* is a strict local minimizer of f . □

Proof. Because $f \in \mathcal{C}^2$, we have $\mathbf{F}(\mathbf{x}^*) = \mathbf{F}^\top(\mathbf{x}^*)$. Using assumption 2 and Rayleigh's inequality it follows that if $\mathbf{d} \neq \mathbf{0}$, then $0 < \lambda_{\min}(\mathbf{F}(\mathbf{x}^*))\|\mathbf{d}\|^2 \leq \mathbf{d}^\top \mathbf{F}(\mathbf{x}^*)\mathbf{d}$. By Taylor's theorem and assumption 1,

$$f(\mathbf{x}^* + \mathbf{d}) - f(\mathbf{x}^*) = \frac{1}{2}\mathbf{d}^\top \mathbf{F}(\mathbf{x}^*)\mathbf{d} + o(\|\mathbf{d}\|^2) \geq \frac{\lambda_{\min}(\mathbf{F}(\mathbf{x}^*))}{2}\|\mathbf{d}\|^2 + o(\|\mathbf{d}\|^2).$$

Hence, for all \mathbf{d} such that $\|\mathbf{d}\|$ is sufficiently small,

$$f(\mathbf{x}^* + \mathbf{d}) > f(\mathbf{x}^*),$$

which completes the proof. ■

Example 6.8 Let $f(\mathbf{x}) = x_1^2 + x_2^2$. We have $\nabla f(\mathbf{x}) = [2x_1, 2x_2]^\top = \mathbf{0}$ if and only if $\mathbf{x} = [0, 0]^\top$. For all $\mathbf{x} \in \mathbb{R}^2$, we have

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} > 0.$$

The point $\mathbf{x} = [0, 0]^\top$ satisfies the FONC, SONC, and SOSC. It is a strict local minimizer. Actually, $\mathbf{x} = [0, 0]^\top$ is a strict global minimizer. Figure 6.8 shows the graph of $f(\mathbf{x}) = x_1^2 + x_2^2$. ■

In this chapter we presented a theoretical basis for the solution of nonlinear unconstrained problems. In the following chapters we are concerned with iterative methods of solving such problems. Such methods are of great importance in practice. Indeed, suppose that one is confronted with a highly nonlinear function of 20 variables. Then, the FONC requires the solution of 20 nonlinear simultaneous equations for 20 variables. These equations, being nonlinear, will normally have multiple solutions. In addition, we would have to compute 210 second derivatives (provided that $f \in \mathcal{C}^2$) to use the SONC or SOSC. We begin our discussion of iterative methods in the next chapter with search methods for functions of one variable.

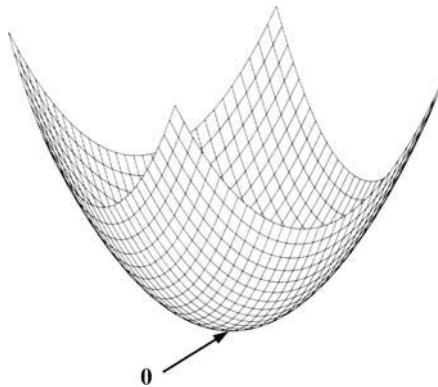


Figure 6.8 Graph of $f(\mathbf{x}) = x_1^2 + x_2^2$.

EXERCISES

6.1 Consider the problem

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}) \\ & \text{subject to} && \mathbf{x} \in \Omega, \end{aligned}$$

where $f \in \mathcal{C}^2$. For each of the following specifications for Ω , \mathbf{x}^* , and f , determine if the given point \mathbf{x}^* is: (i) definitely a local minimizer; (ii) definitely not a local minimizer; or (iii) possibly a local minimizer.

- a. $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, $\Omega = \{\mathbf{x} = [x_1, x_2]^\top : x_1 \geq 1\}$, $\mathbf{x}^* = [1, 2]^\top$, and gradient $\nabla f(\mathbf{x}^*) = [1, 1]^\top$.
- b. $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, $\Omega = \{\mathbf{x} = [x_1, x_2]^\top : x_1 \geq 1, x_2 \geq 2\}$, $\mathbf{x}^* = [1, 2]^\top$, and gradient $\nabla f(\mathbf{x}^*) = [1, 0]^\top$.
- c. $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, $\Omega = \{\mathbf{x} = [x_1, x_2]^\top : x_1 \geq 0, x_2 \geq 0\}$, $\mathbf{x}^* = [1, 2]^\top$, gradient $\nabla f(\mathbf{x}^*) = [0, 0]^\top$, and Hessian $\mathbf{F}(\mathbf{x}^*) = \mathbf{I}$ (identity matrix).
- d. $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, $\Omega = \{\mathbf{x} = [x_1, x_2]^\top : x_1 \geq 1, x_2 \geq 2\}$, $\mathbf{x}^* = [1, 2]^\top$, gradient $\nabla f(\mathbf{x}^*) = [1, 0]^\top$, and Hessian

$$\mathbf{F}(\mathbf{x}^*) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

6.2 Find minimizers and maximizers of the function

$$f(x_1, x_2) = \frac{1}{3}x_1^3 - 4x_1 + \frac{1}{3}x_2^3 - 16x_2.$$

6.3 Show that if \mathbf{x}^* is a global minimizer of f over Ω , and $\mathbf{x}^* \in \Omega' \subset \Omega$, then \mathbf{x}^* is a global minimizer of f over Ω' .

6.4 Suppose that \mathbf{x}^* is a local minimizer of f over Ω , and $\Omega \subset \Omega'$. Show that if \mathbf{x}^* is an interior point of Ω , then \mathbf{x}^* is a local minimizer of f over Ω' . Show that the same conclusion cannot be made if \mathbf{x}^* is not an interior point of Ω .

6.5 Consider the problem of minimizing $f : \mathbb{R} \rightarrow \mathbb{R}$, $f \in \mathcal{C}^3$, over the constraint set Ω . Suppose that 0 is an *interior point* of Ω .

- a. Suppose that 0 is a local minimizer. By the FONC we know that $f'(0) = 0$ (where f' is the first derivative of f). By the SONC we know that $f''(0) \geq 0$ (where f'' is the second derivative of f). State and prove a *third-order necessary condition (TONC)* involving the third derivative at 0, $f'''(0)$.
- b. Give an example of f such that the FONC, SONC, and TONC (in part a) hold at the interior point 0, but 0 is not a local minimizer of f over Ω . (Show that your example is correct.)
- c. Suppose that f is a third-order polynomial. If 0 satisfies the FONC, SONC, and TONC (in part a), then is this *sufficient* for 0 to be a local minimizer?

6.6 Consider the problem of minimizing $f : \mathbb{R} \rightarrow \mathbb{R}$, $f \in \mathcal{C}^3$, over the constraint set $\Omega = [0, 1]$. Suppose that $x^* = 0$ is a local minimizer.

- a. By the FONC we know that $f'(0) \geq 0$ (where f' is the first derivative of f). By the SONC we know that if $f'(0) = 0$, then $f''(0) \geq 0$ (where f'' is the second derivative of f). State and prove a *third-order necessary condition* involving the third derivative at 0, $f'''(0)$.
- b. Give an example of f such that the FONC, SONC, and TONC (in part a) hold at the point 0, but 0 is not a local minimizer of f over $\Omega = [0, 1]$.

6.7 Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $\mathbf{x}_0 \in \mathbb{R}^n$, and $\Omega \subset \mathbb{R}^n$. Show that

$$\mathbf{x}_0 + \arg \min_{\mathbf{x} \in \Omega} f(\mathbf{x}) = \arg \min_{\mathbf{y} \in \Omega'} f(\mathbf{y}),$$

where $\Omega' = \{\mathbf{y} : \mathbf{y} - \mathbf{x}_0 \in \Omega\}$.

6.8 Consider the following function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$:

$$f(\mathbf{x}) = \mathbf{x}^\top \begin{bmatrix} 1 & 2 \\ 4 & 7 \end{bmatrix} \mathbf{x} + \mathbf{x}^\top \begin{bmatrix} 3 \\ 5 \end{bmatrix} + 6.$$

- a. Find the gradient and Hessian of f at the point $[1, 1]^\top$.
- b. Find the directional derivative of f at $[1, 1]^\top$ with respect to a unit vector in the direction of maximal rate of increase.
- c. Find a point that satisfies the FONC (interior case) for f . Does this point satisfy the SONC (for a minimizer)?

6.9 Consider the following function:

$$f(x_1, x_2) = x_1^2 x_2 + x_2^3 x_1.$$

- a. In what direction does the function f decrease most rapidly at the point $\mathbf{x}^{(0)} = [2, 1]^\top$?
- b. What is the rate of increase of f at the point $\mathbf{x}^{(0)}$ in the direction of maximum decrease of f ?
- c. Find the rate of increase of f at the point $\mathbf{x}^{(0)}$ in the direction $\mathbf{d} = [3, 4]^\top$.

6.10 Consider the following function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$:

$$f(\mathbf{x}) = \mathbf{x}^\top \begin{bmatrix} 2 & 5 \\ -1 & 1 \end{bmatrix} \mathbf{x} + \mathbf{x}^\top \begin{bmatrix} 3 \\ 4 \end{bmatrix} + 7.$$

- a. Find the directional derivative of f at $[0, 1]^\top$ in the direction $[1, 0]^\top$.
- b. Find all points that satisfy the first-order necessary condition for f . Does f have a minimizer? If it does, then find all minimizer(s); otherwise, explain why it does not.

6.11 Consider the problem

$$\begin{aligned} &\text{minimize} && -x_2^2 \\ &\text{subject to} && |x_2| \leq x_1^2 \\ & && x_1 \geq 0, \end{aligned}$$

where $x_1, x_2 \in \mathbb{R}$.

- a. Does the point $[x_1, x_2]^\top = \mathbf{0}$ satisfy the first-order necessary condition for a minimizer? That is, if f is the objective function, is it true that $\mathbf{d}^\top \nabla f(\mathbf{0}) \geq 0$ for all feasible directions \mathbf{d} at $\mathbf{0}$?
- b. Is the point $[x_1, x_2]^\top = \mathbf{0}$ a local minimizer, a strict local minimizer, a local maximizer, a strict local maximizer, or none of the above?

6.12 Consider the problem

$$\begin{aligned} & \text{minimize } f(\mathbf{x}) \\ & \text{subject to } \mathbf{x} \in \Omega, \end{aligned}$$

where $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ is given by $f(\mathbf{x}) = 5x_2$ with $\mathbf{x} = [x_1, x_2]^\top$, and $\Omega = \{\mathbf{x} = [x_1, x_2]^\top : x_1^2 + x_2 \geq 1\}$.

- a. Does the point $\mathbf{x}^* = [0, 1]^\top$ satisfy the first-order necessary condition?
- b. Does the point $\mathbf{x}^* = [0, 1]^\top$ satisfy the second-order necessary condition?
- c. Is the point $\mathbf{x}^* = [0, 1]^\top$ a local minimizer?

6.13 Consider the problem

$$\begin{aligned} & \text{minimize } f(\mathbf{x}) \\ & \text{subject to } \mathbf{x} \in \Omega, \end{aligned}$$

where $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ is given by $f(\mathbf{x}) = -3x_1$ with $\mathbf{x} = [x_1, x_2]^\top$, and $\Omega = \{\mathbf{x} = [x_1, x_2]^\top : x_1 + x_2^2 \leq 2\}$. Answer each of the following questions, showing complete justification.

- a. Does the point $\mathbf{x}^* = [2, 0]^\top$ satisfy the first-order necessary condition?
- b. Does the point $\mathbf{x}^* = [2, 0]^\top$ satisfy the second-order necessary condition?
- c. Is the point $\mathbf{x}^* = [2, 0]^\top$ a local minimizer?

6.14 Consider the problem

$$\begin{aligned} & \text{minimize } f(\mathbf{x}) \\ & \text{subject to } \mathbf{x} \in \Omega, \end{aligned}$$

where $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : x_1^2 + x_2^2 \geq 1\}$ and $f(\mathbf{x}) = x_2$.

- a. Find all point(s) satisfying the FONC.
- b. Which of the point(s) in part a satisfy the SONC?
- c. Which of the point(s) in part a are local minimizers?

6.15 Consider the problem

$$\begin{aligned} & \text{minimize } f(\mathbf{x}) \\ & \text{subject to } \mathbf{x} \in \Omega \end{aligned}$$

where $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ is given by $f(\mathbf{x}) = 3x_1$ with $\mathbf{x} = [x_1, x_2]^\top$, and $\Omega = \{\mathbf{x} = [x_1, x_2]^\top : x_1 + x_2^2 \geq 2\}$. Answer each of the following questions, showing complete justification.

- Does the point $\mathbf{x}^* = [2, 0]^\top$ satisfy the first-order necessary condition?
- Does the point $\mathbf{x}^* = [2, 0]^\top$ satisfy the second-order necessary condition?
- Is the point $\mathbf{x}^* = [2, 0]^\top$ a local minimizer?

Hint: Draw a picture with the constraint set and level sets of f .

6.16 Consider the problem

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}) \\ & \text{subject to} && \mathbf{x} \in \Omega, \end{aligned}$$

where $\mathbf{x} = [x_1, x_2]^\top$, $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ is given by $f(\mathbf{x}) = 4x_1^2 - x_2^2$, and $\Omega = \{\mathbf{x} : x_1^2 + 2x_1 - x_2 \geq 0, x_1 \geq 0, x_2 \geq 0\}$.

- Does the point $\mathbf{x}^* = \mathbf{0} = [0, 0]^\top$ satisfy the first-order necessary condition?
- Does the point $\mathbf{x}^* = \mathbf{0}$ satisfy the second-order necessary condition?
- Is the point $\mathbf{x}^* = \mathbf{0}$ a local minimizer of the given problem?

6.17 Consider the problem

$$\begin{aligned} & \text{maximize} && f(\mathbf{x}) \\ & \text{subject to} && \mathbf{x} \in \Omega, \end{aligned}$$

where $\Omega \subset \{\mathbf{x} \in \mathbb{R}^2 : x_1 > 0, x_2 > 0\}$ and $f : \Omega \rightarrow \mathbb{R}$ is given by $f(\mathbf{x}) = \log(x_1) + \log(x_2)$ with $\mathbf{x} = [x_1, x_2]^\top$, where “log” represents natural logarithm. Suppose that \mathbf{x}^* is an optimal solution. Answer each of the following questions, showing complete justification.

- Is it possible that \mathbf{x}^* is an interior point of Ω ?
- At what point(s) (if any) is the second-order necessary condition satisfied?

6.18 Suppose that we are given n real numbers, x_1, \dots, x_n . Find the number $\bar{x} \in \mathbb{R}$ such that the sum of the squared difference between \bar{x} and the numbers above is minimized (assuming that the solution \bar{x} exists).

6.19 An art collector stands at a distance of x feet from the wall, where a piece of art (picture) of height a feet is hung, b feet above his eyes, as shown in

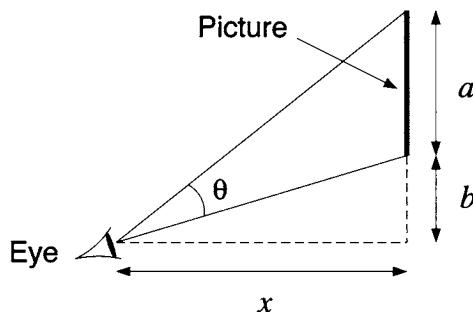


Figure 6.9 Art collector's eye position in Exercise 6.19.

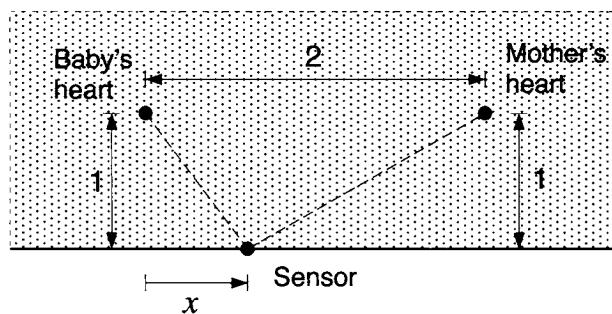


Figure 6.10 Simplified fetal heart monitoring system for Exercise 6.20.

Figure 6.9. Find the distance from the wall for which the angle θ subtended by the eye to the picture is maximized.

Hint: (1) Maximizing θ is equivalent to maximizing $\tan(\theta)$.

(2) If $\theta = \theta_2 - \theta_1$, then $\tan(\theta) = (\tan(\theta_2) - \tan(\theta_1))/(1 + \tan(\theta_2)\tan(\theta_1))$.

6.20 Figure 6.10 shows a simplified model of a fetal heart monitoring system (the distances shown have been scaled down to make the calculations simpler). A heartbeat sensor is located at position x (see Figure 6.10).

The energy of the heartbeat signal measured by the sensor is the reciprocal of the squared distance from the source (baby's heart or mother's heart). Find the position of the sensor that maximizes the *signal-to-interference ratio*, which is the ratio of the signal energy from the baby's heart to the signal energy from the mother's heart.

6.21 An amphibian vehicle needs to travel from point A (on land) to point B (in water), as illustrated in Figure 6.11. The speeds at which the vehicle travels on land and water are v_1 and v_2 , respectively.

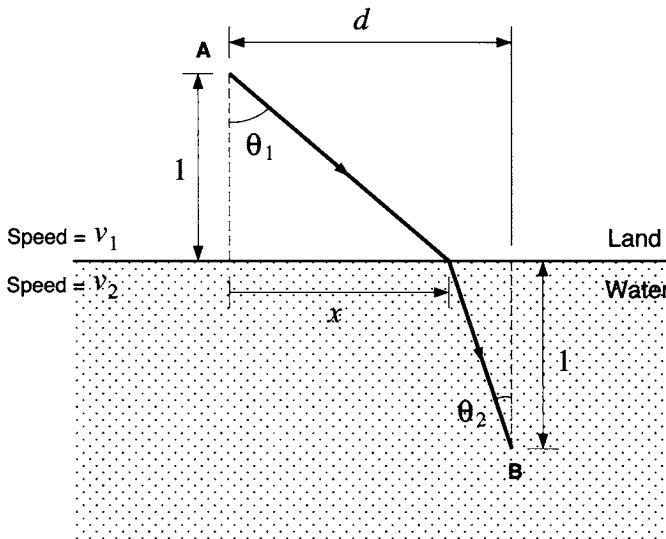


Figure 6.11 Path of amphibian vehicle in Exercise 6.21.

- a. Suppose that the vehicle traverses a path that minimizes the total time taken to travel from A to B. Use the first-order necessary condition to show that for the optimal path above, the angles θ_1 and θ_2 in Figure 6.11 satisfy Snell's law:

$$\frac{\sin \theta_1}{\sin \theta_2} = \frac{v_1}{v_2}.$$

- b. Does the minimizer for the problem in part a satisfy the second-order sufficient condition?

6.22 Suppose that you have a piece of land to sell and you have two buyers. If the first buyer receives a fraction x_1 of the piece of land, the buyer will pay you $U_1(x_1)$ dollars. Similarly, the second buyer will pay you $U_2(x_2)$ dollars for a fraction of x_2 of the land. Your goal is to sell parts of your land to the two buyers so that you maximize the total dollars you receive. (Other than the constraint that you can only sell whatever land you own, there are no restrictions on how much land you can sell to each buyer.)

- a. Formulate the problem as an optimization problem of the kind

$$\begin{aligned} & \text{maximize } f(\mathbf{x}) \\ & \text{subject to } \mathbf{x} \in \Omega \end{aligned}$$

by specifying f and Ω . Draw a picture of the constraint set.

- b. Suppose that $U_i(x_i) = a_i x_i$, $i = 1, 2$, where a_1 and a_2 are given positive constants such that $a_1 > a_2$. Find all feasible points that satisfy the first-order necessary condition, giving full justification.
- c. Among those points in the answer of part b, find all that also satisfy the second-order necessary condition.

6.23 Let $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ be defined by

$$f(\mathbf{x}) = (x_1 - x_2)^4 + x_1^2 - x_2^2 - 2x_1 + 2x_2 + 1,$$

where $\mathbf{x} = [x_1, x_2]^\top$. Suppose that we wish to minimize f over \mathbb{R}^2 . Find all points satisfying the FONC. Do these points satisfy the SONC?

6.24 Show that if \mathbf{d} is a feasible direction at a point $\mathbf{x} \in \Omega$, then for all $\beta > 0$, the vector $\beta\mathbf{d}$ is also a feasible direction at \mathbf{x} .

6.25 Let $\Omega = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b}\}$. Show that $\mathbf{d} \in \mathbb{R}^n$ is a feasible direction at $\mathbf{x} \in \Omega$ if and only if $\mathbf{A}\mathbf{d} = \mathbf{0}$.

6.26 Let $f : \mathbb{R}^2 \rightarrow \mathbb{R}$. Consider the problem

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && x_1, x_2 \geq 0, \end{aligned}$$

where $\mathbf{x} = [x_1, x_2]^\top$. Suppose that $\nabla f(\mathbf{0}) \neq \mathbf{0}$, and

$$\frac{\partial f}{\partial x_1}(\mathbf{0}) \leq 0, \quad \frac{\partial f}{\partial x_2}(\mathbf{0}) \leq 0.$$

Show that $\mathbf{0}$ cannot be a minimizer for this problem.

6.27 Let $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{c} \neq \mathbf{0}$, and consider the problem of minimizing the function $f(\mathbf{x}) = \mathbf{c}^\top \mathbf{x}$ over a constraint set $\Omega \subset \mathbb{R}^n$. Show that we cannot have a solution lying in the interior of Ω .

6.28 Consider the problem

$$\begin{aligned} &\text{maximize} && c_1 x_1 + c_2 x_2 \\ &\text{subject to} && x_1 + x_2 \leq 1 \\ & && x_1, x_2 \geq 0, \end{aligned}$$

where c_1 and c_2 are constants such that $c_1 > c_2 \geq 0$. This is a *linear programming* problem (see Part III). Assuming that the problem has an optimal feasible solution, use the first-order necessary condition to show that the *unique* optimal feasible solution \mathbf{x}^* is $[1, 0]^\top$.

Hint: First show that \mathbf{x}^* cannot lie in the interior of the constraint set. Then, show that \mathbf{x}^* cannot lie on the line segments $L_1 = \{\mathbf{x} : x_1 = 0, 0 \leq x_2 < 1\}$, $L_2 = \{\mathbf{x} : 0 \leq x_1 < 1, x_2 = 0\}$, $L_3 = \{\mathbf{x} : 0 \leq x_1 < 1, x_2 = 1 - x_1\}$.

6.29 Line Fitting. Let $[x_1, y_1]^\top, \dots, [x_n, y_n]^\top$, $n \geq 2$, be points on the \mathbb{R}^2 plane (each $x_i, y_i \in \mathbb{R}$). We wish to find the straight line of “best fit” through these points (“best” in the sense that the average squared error is minimized); that is, we wish to find $a, b \in \mathbb{R}$ to minimize

$$f(a, b) = \frac{1}{n} \sum_{i=1}^n (ax_i + b - y_i)^2.$$

a. Let

$$\bar{X} = \frac{1}{n} \sum_{i=1}^n x_i,$$

$$\bar{Y} = \frac{1}{n} \sum_{i=1}^n y_i,$$

$$\bar{X^2} = \frac{1}{n} \sum_{i=1}^n x_i^2,$$

$$\bar{Y^2} = \frac{1}{n} \sum_{i=1}^n y_i^2,$$

$$\bar{XY} = \frac{1}{n} \sum_{i=1}^n x_i y_i.$$

Show that $f(a, b)$ can be written in the form $\mathbf{z}^\top \mathbf{Q} \mathbf{z} - 2\mathbf{c}^\top \mathbf{z} + d$, where $\mathbf{z} = [a, b]^\top$, $\mathbf{Q} = \mathbf{Q}^\top \in \mathbb{R}^{2 \times 2}$, $\mathbf{c} \in \mathbb{R}^2$ and $d \in \mathbb{R}$, and find expressions for \mathbf{Q} , \mathbf{c} , and d in terms of \bar{X} , \bar{Y} , $\bar{X^2}$, $\bar{Y^2}$, and \bar{XY} .

b. Assume that the x_i , $i = 1, \dots, n$, are not all equal. Find the parameters a^* and b^* for the line of best fit in terms of \bar{X} , \bar{Y} , $\bar{X^2}$, $\bar{Y^2}$, and \bar{XY} . Show that the point $[a^*, b^*]^\top$ is the only local minimizer of f .

Hint: $\bar{X^2} - (\bar{X})^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{X})^2$.

c. Show that if a^* and b^* are the parameters of the line of best fit, then $\bar{Y} = a^* \bar{X} + b^*$ (and hence once we have computed a^* , we can compute b^* using the formula $b^* = \bar{Y} - a^* \bar{X}$).

6.30 Suppose that we are given a set of vectors $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(p)}\}$, $\mathbf{x}^{(i)} \in \mathbb{R}^n$, $i = 1, \dots, p$. Find the vector $\bar{\mathbf{x}} \in \mathbb{R}^n$ such that the average squared distance (norm) between $\bar{\mathbf{x}}$ and $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(p)}$,

$$\frac{1}{p} \sum_{i=1}^p \|\bar{\mathbf{x}} - \mathbf{x}^{(i)}\|^2,$$

is minimized. Use the SOSC to prove that the vector $\bar{\mathbf{x}}$ found above is a strict local minimizer. How is $\bar{\mathbf{x}}$ related to the centroid (or center of gravity) of the given set of points $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(p)}\}$?

6.31 Consider a function $f : \Omega \rightarrow \mathbb{R}$, where $\Omega \subset \mathbb{R}^n$ is a convex set and $f \in \mathcal{C}^1$. Given $\mathbf{x}^* \in \Omega$, suppose that there exists $c > 0$ such that $\mathbf{d}^\top \nabla f(\mathbf{x}^*) \geq c\|\mathbf{d}\|$ for all feasible directions \mathbf{d} at \mathbf{x}^* . Show that \mathbf{x}^* is a strict local minimizer of f over Ω .

6.32 Prove the following generalization of the second-order sufficient condition:

Theorem: Let Ω be a convex subset of \mathbb{R}^n , $f \in \mathcal{C}^2$ a real-valued function on Ω , and \mathbf{x}^* a point in Ω . Suppose that there exists $c \in \mathbb{R}$, $c > 0$, such that for all feasible directions \mathbf{d} at \mathbf{x}^* ($\mathbf{d} \neq \mathbf{0}$), the following hold:

1. $\mathbf{d}^\top \nabla f(\mathbf{x}^*) \geq 0$.
2. $\mathbf{d}^\top \mathbf{F}(\mathbf{x}^*)\mathbf{d} \geq c\|\mathbf{d}\|^2$.

Then, \mathbf{x}^* is a strict local minimizer of f .

6.33 Consider the quadratic function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ given by

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b},$$

where $\mathbf{Q} = \mathbf{Q}^\top > 0$. Show that \mathbf{x}^* minimizes f if and only if \mathbf{x}^* satisfies the FONC.

6.34 Consider the linear system $x_{k+1} = ax_k + bu_{k+1}$, $k \geq 0$, where $x_i \in \mathbb{R}$, $u_i \in \mathbb{R}$, and the initial condition is $x_0 = 0$. Find the values of the control inputs u_1, \dots, u_n to minimize

$$-qx_n + r \sum_{i=1}^n u_i^2,$$

where $q, r > 0$ are given constants. This can be interpreted as desiring to make x_n as large as possible but at the same time desiring to make the total input energy $\sum_{i=1}^n u_i^2$ as small as possible. The constants q and r reflect the relative weights of these two objectives.

CHAPTER 7

ONE-DIMENSIONAL SEARCH METHODS

7.1 Introduction

In this chapter, we are interested in the problem of minimizing an objective function $f : \mathbb{R} \rightarrow \mathbb{R}$ (i.e., a one-dimensional problem). The approach is to use an iterative search algorithm, also called a line-search method. One-dimensional search methods are of interest for the following reasons. First, they are special cases of search methods used in multivariable problems. Second, they are used as part of general multivariable algorithms (as described later in Section 7.8).

In an iterative algorithm, we start with an initial candidate solution $x^{(0)}$ and generate a sequence of *iterates* $x^{(1)}, x^{(2)}, \dots$. For each iteration $k = 0, 1, 2, \dots$, the next point $x^{(k+1)}$ depends on $x^{(k)}$ and the objective function f . The algorithm may use only the value of f at specific points, or perhaps its first derivative f' , or even its second derivative f'' . In this chapter, we study several algorithms:

- Golden section method (uses only f)
- Fibonacci method (uses only f)

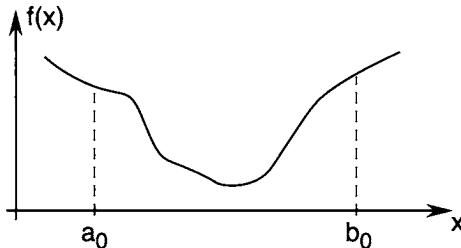


Figure 7.1 Unimodal function.

- Bisection method (uses only f')
- Secant method (uses only f')
- Newton's method (uses f' and f'')

The exposition here is based on [27].

7.2 Golden Section Search

The search methods we discuss in this and the next two sections allow us to determine the minimizer of an objective function $f : \mathbb{R} \rightarrow \mathbb{R}$ over a closed interval, say $[a_0, b_0]$. The only property that we assume of the objective function f is that it is *unimodal*, which means that f has only one local minimizer. An example of such a function is depicted in Figure 7.1.

The methods we discuss are based on evaluating the objective function at different points in the interval $[a_0, b_0]$. We choose these points in such a way that an approximation to the minimizer of f may be achieved in as few evaluations as possible. Our goal is to narrow the range progressively until the minimizer is “boxed in” with sufficient accuracy.

Consider a unimodal function f of one variable and the interval $[a_0, b_0]$. If we evaluate f at only one intermediate point of the interval, we cannot narrow the range within which we know the minimizer is located. We have to evaluate f at two intermediate points, as illustrated in Figure 7.2. We choose the intermediate points in such a way that the reduction in the range is symmetric, in the sense that

$$a_1 - a_0 = b_0 - b_1 = \rho(b_0 - a_0),$$

where

$$\rho < \frac{1}{2}.$$

We then evaluate f at the intermediate points. If $f(a_1) < f(b_1)$, then the minimizer must lie in the range $[a_0, b_1]$. If, on the other hand, $f(a_1) \geq f(b_1)$, then the minimizer is located in the range $[a_1, b_0]$ (see Figure 7.3).

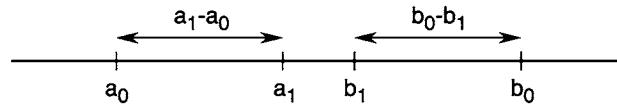


Figure 7.2 Evaluating the objective function at two intermediate points.

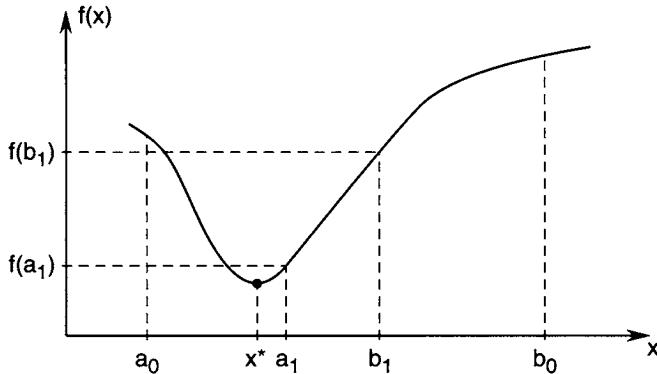


Figure 7.3 The case where $f(a_1) < f(b_1)$; the minimizer $x^* \in [a_0, b_1]$.

Starting with the reduced range of uncertainty, we can repeat the process and similarly find two new points, say a_2 and b_2 , using the same value of $\rho < \frac{1}{2}$ as before. However, we would like to minimize the number of objective function evaluations while reducing the width of the uncertainty interval. Suppose, for example, that $f(a_1) < f(b_1)$, as in Figure 7.3. Then, we know that $x^* \in [a_0, b_1]$. Because a_1 is already in the uncertainty interval and $f(a_1)$ is already known, we can make a_1 coincide with b_2 . Thus, only one new evaluation of f at a_2 would be necessary. To find the value of ρ that results in only one new evaluation of f , see Figure 7.4. Without loss of generality, imagine that the original range $[a_0, b_0]$ is of unit length. Then, to have only one new evaluation of f it is enough to choose ρ so that

$$\rho(b_1 - a_0) = b_1 - b_2.$$

Because $b_1 - a_0 = 1 - \rho$ and $b_1 - b_2 = 1 - 2\rho$, we have

$$\rho(1 - \rho) = 1 - 2\rho.$$

We write the quadratic equation above as

$$\rho^2 - 3\rho + 1 = 0.$$

The solutions are

$$\rho_1 = \frac{3 + \sqrt{5}}{2}, \quad \rho_2 = \frac{3 - \sqrt{5}}{2}.$$

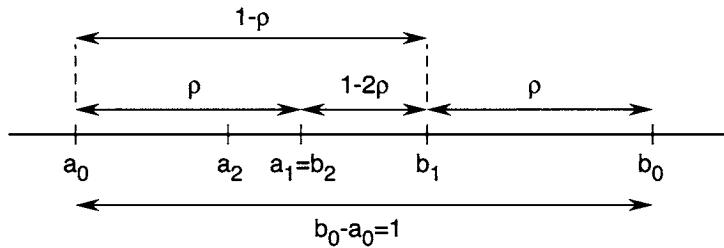


Figure 7.4 Finding value of ρ resulting in only one new evaluation of f .

Because we require that $\rho < \frac{1}{2}$, we take

$$\rho = \frac{3 - \sqrt{5}}{2} \approx 0.382.$$

Observe that

$$1 - \rho = \frac{\sqrt{5} - 1}{2}$$

and

$$\frac{\rho}{1 - \rho} = \frac{3 - \sqrt{5}}{\sqrt{5} - 1} = \frac{\sqrt{5} - 1}{2} = \frac{1 - \rho}{1},$$

that is,

$$\frac{\rho}{1 - \rho} = \frac{1 - \rho}{1}.$$

Thus, dividing a range in the ratio of ρ to $1 - \rho$ has the effect that the ratio of the shorter segment to the longer equals the ratio of the longer to the sum of the two. This rule was referred to by ancient Greek geometers as the *golden section*.

Using the golden section rule means that at every stage of the uncertainty range reduction (except the first), the objective function f need only be evaluated at one new point. The uncertainty range is reduced by the ratio $1 - \rho \approx 0.61803$ at every stage. Hence, N steps of reduction using the golden section method reduces the range by the factor

$$(1 - \rho)^N \approx (0.61803)^N.$$

Example 7.1 Suppose that we wish to use the golden section search method to find the value of x that minimizes

$$f(x) = x^4 - 14x^3 + 60x^2 - 70x$$

in the interval $[0, 2]$ (this function comes from an example in [21]). We wish to locate this value of x to within a range of 0.3.

After N stages the range $[0, 2]$ is reduced by $(0.61803)^N$. So, we choose N so that

$$(0.61803)^N \leq 0.3/2.$$

Four stages of reduction will do; that is, $N = 4$.

Iteration 1. We evaluate f at two intermediate points a_1 and b_1 . We have

$$\begin{aligned} a_1 &= a_0 + \rho(b_0 - a_0) = 0.7639, \\ b_1 &= a_0 + (1 - \rho)(b_0 - a_0) = 1.236, \end{aligned}$$

where $\rho = (3 - \sqrt{5})/2$. We compute

$$\begin{aligned} f(a_1) &= -24.36, \\ f(b_1) &= -18.96. \end{aligned}$$

Thus, $f(a_1) < f(b_1)$, so the uncertainty interval is reduced to

$$[a_0, b_1] = [0, 1.236].$$

Iteration 2. We choose b_2 to coincide with a_1 , and so f need only be evaluated at one new point,

$$a_2 = a_0 + \rho(b_1 - a_0) = 0.4721.$$

We have

$$\begin{aligned} f(a_2) &= -21.10, \\ f(b_2) &= f(a_1) = -24.36. \end{aligned}$$

Now, $f(b_2) < f(a_2)$, so the uncertainty interval is reduced to

$$[a_2, b_1] = [0.4721, 1.236].$$

Iteration 3. We set $a_3 = b_2$ and compute b_3 :

$$b_3 = a_2 + (1 - \rho)(b_1 - a_2) = 0.9443.$$

We have

$$\begin{aligned} f(a_3) &= f(b_2) = -24.36, \\ f(b_3) &= -23.59. \end{aligned}$$

So $f(b_3) > f(a_3)$. Hence, the uncertainty interval is further reduced to

$$[a_2, b_3] = [0.4721, 0.9443].$$

Iteration 4. We set $b_4 = a_3$ and

$$a_4 = a_2 + \rho(b_3 - a_2) = 0.6525.$$

We have

$$\begin{aligned} f(a_4) &= -23.84, \\ f(b_4) = f(a_3) &= -24.36. \end{aligned}$$

Hence, $f(a_4) > f(b_4)$. Thus, the value of x that minimizes f is located in the interval

$$[a_4, b_3] = [0.6525, 0.9443].$$

Note that $b_3 - a_4 = 0.292 < 0.3$. ■

7.3 Fibonacci Method

Recall that the golden section method uses the same value of ρ throughout. Suppose now that we are allowed to vary the value ρ from stage to stage, so that at the k th stage in the reduction process we use a value ρ_k , at the next stage we use a value ρ_{k+1} , and so on.

As in the golden section search, our goal is to select successive values of ρ_k , $0 \leq \rho_k \leq 1/2$, such that only one new function evaluation is required at each stage. To derive the strategy for selecting evaluation points, consider Figure 7.5. From this figure we see that it is sufficient to choose the ρ_k such that

$$\rho_{k+1}(1 - \rho_k) = 1 - 2\rho_k.$$

After some manipulations, we obtain

$$\rho_{k+1} = 1 - \frac{\rho_k}{1 - \rho_k}.$$

There are many sequences ρ_1, ρ_2, \dots that satisfy the law of formation above and the condition that $0 \leq \rho_k \leq 1/2$. For example, the sequence $\rho_1 = \rho_2 = \rho_3 = \dots = (3 - \sqrt{5})/2$ satisfies the conditions above and gives rise to the golden section method.

Suppose that we are given a sequence ρ_1, ρ_2, \dots satisfying the conditions above and we use this sequence in our search algorithm. Then, after N iterations of the algorithm, the uncertainty range is reduced by a factor of

$$(1 - \rho_1)(1 - \rho_2) \cdots (1 - \rho_N).$$

Depending on the sequence ρ_1, ρ_2, \dots , we get a different reduction factor. The natural question is as follows: What sequence ρ_1, ρ_2, \dots minimizes the reduction factor above? This problem is a constrained optimization problem that can be stated formally as

$$\begin{aligned} &\text{minimize} && (1 - \rho_1)(1 - \rho_2) \cdots (1 - \rho_N) \\ &\text{subject to} && \rho_{k+1} = 1 - \frac{\rho_k}{1 - \rho_k}, \quad k = 1, \dots, N-1 \\ & && 0 \leq \rho_k \leq \frac{1}{2}, \quad k = 1, \dots, N. \end{aligned}$$

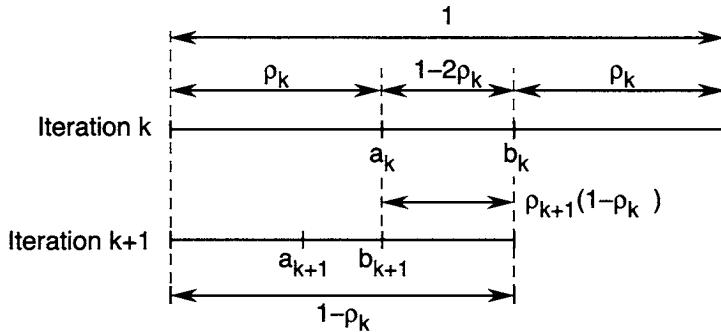


Figure 7.5 Selecting evaluation points.

Before we give the solution to the optimization problem above, we need to introduce the *Fibonacci sequence* F_1, F_2, F_3, \dots . This sequence is defined as follows. First, let $F_{-1} = 0$ and $F_0 = 1$ by convention. Then, for $k \geq 1$,

$$F_{k+1} = F_k + F_{k-1}.$$

Some values of elements in the Fibonacci sequence are:

F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
1	2	3	5	8	13	21	34

It turns out that the solution to the optimization problem above is

$$\rho_1 = 1 - \frac{F_N}{F_{N+1}},$$

$$\rho_2 = 1 - \frac{F_{N-1}}{F_N},$$

⋮

$$\rho_k = 1 - \frac{F_{N-k+1}}{F_{N-k+2}},$$

⋮

$$\rho_N = 1 - \frac{F_1}{F_2},$$

where the F_k are the elements of the Fibonacci sequence. The resulting algorithm is called the *Fibonacci search method*. We present a proof for the optimality of the Fibonacci search method later in this section.

In the Fibonacci search method, the uncertainty range is reduced by the factor

$$(1 - \rho_1)(1 - \rho_2) \cdots (1 - \rho_N) = \frac{F_N}{F_{N+1}} \frac{F_{N-1}}{F_N} \cdots \frac{F_1}{F_2} = \frac{F_1}{F_{N+1}} = \frac{1}{F_{N+1}}.$$

Because the Fibonacci method uses the optimal values of ρ_1, ρ_2, \dots , the reduction factor above is less than that of the golden section method. In other words, the Fibonacci method is better than the golden section method in that it gives a smaller final uncertainty range.

We point out that there is an anomaly in the final iteration of the Fibonacci search method, because

$$\rho_N = 1 - \frac{F_1}{F_2} = \frac{1}{2}.$$

Recall that we need two intermediate points at each stage, one that comes from a previous iteration and another that is a new evaluation point. However, with $\rho_N = 1/2$, the two intermediate points coincide in the middle of the uncertainty interval, and therefore we cannot further reduce the uncertainty range. To get around this problem, we perform the new evaluation for the last iteration using $\rho_N = 1/2 - \varepsilon$, where ε is a small number. In other words, the new evaluation point is just to the left or right of the midpoint of the uncertainty interval. This modification to the Fibonacci method is, of course, of no significant practical consequence.

As a result of the modification above, the reduction in the uncertainty range at the last iteration may be either

$$1 - \rho_N = \frac{1}{2}$$

or

$$1 - (\rho_N - \varepsilon) = \frac{1}{2} + \varepsilon = \frac{1 + 2\varepsilon}{2},$$

depending on which of the two points has the smaller objective function value. Therefore, in the worst case, the reduction factor in the uncertainty range for the Fibonacci method is

$$\frac{1 + 2\varepsilon}{F_{N+1}}.$$

Example 7.2 Consider the function

$$f(x) = x^4 - 14x^3 + 60x^2 - 70x.$$

Suppose that we wish to use the Fibonacci search method to find the value of x that minimizes f over the range $[0, 2]$, and locate this value of x to within the range 0.3.

After N steps the range is reduced by $(1 + 2\varepsilon)/F_{N+1}$ in the worst case. We need to choose N such that

$$\frac{1 + 2\varepsilon}{F_{N+1}} \leq \frac{\text{final range}}{\text{initial range}} = \frac{0.3}{2} = 0.15.$$

Thus, we need

$$F_{N+1} \geq \frac{1 + 2\varepsilon}{0.15}.$$

If we choose $\varepsilon \leq 0.1$, then $N = 4$ will do.

Iteration 1. We start with

$$1 - \rho_1 = \frac{F_4}{F_5} = \frac{5}{8}.$$

We then compute

$$\begin{aligned} a_1 &= a_0 + \rho_1(b_0 - a_0) = \frac{3}{4}, \\ b_1 &= a_0 + (1 - \rho_1)(b_0 - a_0) = \frac{5}{4}, \\ f(a_1) &= -24.34, \\ f(b_1) &= -18.65, \\ f(a_1) &< f(b_1). \end{aligned}$$

The range is reduced to

$$[a_0, b_1] = \left[0, \frac{5}{4}\right].$$

Iteration 2. We have

$$\begin{aligned} 1 - \rho_2 &= \frac{F_3}{F_4} = \frac{3}{5}, \\ a_2 &= a_0 + \rho_2(b_1 - a_0) = \frac{1}{2}, \\ b_2 &= a_1 = \frac{3}{4}, \\ f(a_2) &= -21.69, \\ f(b_2) &= f(a_1) = -24.34, \\ f(a_2) &> f(b_2), \end{aligned}$$

so the range is reduced to

$$[a_2, b_1] = \left[\frac{1}{2}, \frac{5}{4}\right].$$

Iteration 3. We compute

$$\begin{aligned} 1 - \rho_3 &= \frac{F_2}{F_3} = \frac{2}{3}, \\ a_3 &= b_2 = \frac{3}{4}, \\ b_3 &= a_2 + (1 - \rho_3)(b_1 - a_2) = 1, \\ f(a_3) &= f(b_2) = -24.34, \\ f(b_3) &= -23, \\ f(a_3) &< f(b_3). \end{aligned}$$

The range is reduced to

$$[a_2, b_3] = \left[\frac{1}{2}, 1 \right].$$

Iteration 4. We choose $\varepsilon = 0.05$. We have

$$\begin{aligned} 1 - \rho_4 &= \frac{F_1}{F_2} = \frac{1}{2}, \\ a_4 &= a_2 + (\rho_4 - \varepsilon)(b_3 - a_2) = 0.725, \\ b_4 &= a_3 = \frac{3}{4}, \\ f(a_4) &= -24.27, \\ f(b_4) &= f(a_3) = -24.34, \\ f(a_4) &> f(b_4). \end{aligned}$$

The range is reduced to

$$[a_4, b_3] = [0.725, 1].$$

Note that $b_3 - a_4 = 0.275 < 0.3$. ■

We now turn to a proof of the optimality of the Fibonacci search method. Skipping the rest of this section does not affect the continuity of the presentation.

To begin, recall that we wish to prove that the values of $\rho_1, \rho_2, \dots, \rho_N$ used in the Fibonacci method, where $\rho_k = 1 - F_{N-k+1}/F_{N-k+2}$, solve the optimization problem

$$\begin{aligned} &\text{minimize} \quad (1 - \rho_1)(1 - \rho_2) \cdots (1 - \rho_N) \\ &\text{subject to} \quad \rho_{k+1} = 1 - \frac{\rho_k}{1 - \rho_k}, \quad k = 1, \dots, N-1 \\ &\qquad \qquad \qquad 0 \leq \rho_k \leq \frac{1}{2}, \quad k = 1, \dots, N. \end{aligned}$$

It is easy to check that the values of ρ_1, ρ_2, \dots above for the Fibonacci search method satisfy the feasibility conditions in the optimization problem above (see Exercise 7.4). Recall that the Fibonacci method has an overall reduction factor of $(1 - \rho_1) \cdots (1 - \rho_N) = 1/F_{N+1}$. To prove that the Fibonacci search method is optimal, we show that for any feasible values of ρ_1, \dots, ρ_N , we have $(1 - \rho_1) \cdots (1 - \rho_N) \geq 1/F_{N+1}$.

It is more convenient to work with $r_k = 1 - \rho_k$ rather than ρ_k . The optimization problem stated in terms of r_k is

$$\begin{aligned} &\text{minimize} \quad r_1 \cdots r_N \\ &\text{subject to} \quad r_{k+1} = \frac{1}{r_k} - 1, \quad k = 1, \dots, N-1 \\ &\qquad \qquad \qquad \frac{1}{2} \leq r_k \leq 1, \quad k = 1, \dots, N. \end{aligned}$$

Note that if r_1, r_2, \dots satisfy $r_{k+1} = \frac{1}{r_k} - 1$, then $r_k \geq 1/2$ if and only if $r_{k+1} \leq 1$. Also, $r_k \geq 1/2$ if and only if $r_{k-1} \leq 2/3 \leq 1$. Therefore, in the constraints above, we may remove the constraint $r_k \leq 1$, because it is implied implicitly by $r_k \geq 1/2$ and the other constraints. Therefore, the constraints above reduce to

$$\begin{aligned} r_{k+1} &= \frac{1}{r_k} - 1, \quad k = 1, \dots, N-1, \\ r_k &\geq \frac{1}{2}, \quad k = 1, \dots, N. \end{aligned}$$

To proceed, we need the following technical lemmas. In the statements of the lemmas, we assume that r_1, r_2, \dots is a sequence that satisfies

$$r_{k+1} = \frac{1}{r_k} - 1, \quad r_k \geq \frac{1}{2}, \quad k = 1, 2, \dots.$$

Lemma 7.1 *For $k \geq 2$,*

$$r_k = -\frac{F_{k-2} - F_{k-1}r_1}{F_{k-3} - F_{k-2}r_1}.$$

□

Proof. We proceed by induction. For $k = 2$ we have

$$r_2 = \frac{1}{r_1} - 1 = \frac{1 - r_1}{r_1} = -\frac{F_0 - F_1r_1}{F_{-1} - F_0r_1}$$

and hence the lemma holds for $k = 2$. Suppose now that the lemma holds for $k \geq 2$. We show that it also holds for $k + 1$. We have

$$\begin{aligned} r_{k+1} &= \frac{1}{r_k} - 1 \\ &= \frac{-F_{k-3} + F_{k-2}r_1}{F_{k-2} - F_{k-1}r_1} - \frac{F_{k-2} - F_{k-1}r_1}{F_{k-2} - F_{k-1}r_1} \\ &= -\frac{F_{k-2} + F_{k-3} - (F_{k-1} + F_{k-2})r_1}{F_{k-2} - F_{k-1}r_1} \\ &= -\frac{F_{k-1} - F_kr_1}{F_{k-2} - F_{k-1}r_1}, \end{aligned}$$

where we used the formation law for the Fibonacci sequence. ■

Lemma 7.2 *For $k \geq 2$,*

$$(-1)^k(F_{k-2} - F_{k-1}r_1) > 0.$$

□

Proof. We proceed by induction. For $k = 2$, we have

$$(-1)^2(F_0 - F_1 r_1) = 1 - r_1.$$

But $r_1 = 1/(1 + r_2) \leq 2/3$, and hence $1 - r_1 > 0$. Therefore, the result holds for $k = 2$. Suppose now that the lemma holds for $k \geq 2$. We show that it also holds for $k + 1$. We have

$$(-1)^{k+1}(F_{k-1} - F_k r_1) = (-1)^{k+1}r_{k+1} \frac{1}{r_{k+1}}(F_{k-1} - F_k r_1).$$

By Lemma 7.1,

$$r_{k+1} = -\frac{F_{k-1} - F_k r_1}{F_{k-2} - F_{k-1} r_1}.$$

Substituting for $1/r_{k+1}$, we obtain

$$(-1)^{k+1}(F_{k-1} - F_k r_1) = r_{k+1}(-1)^k(F_{k-2} - F_{k-1} r_1) > 0,$$

which completes the proof. ■

Lemma 7.3 *For $k \geq 2$,*

$$(-1)^{k+1}r_1 \geq (-1)^{k+1} \frac{F_k}{F_{k+1}}.$$

□

Proof. Because $r_{k+1} = \frac{1}{r_k} - 1$ and $r_k \geq \frac{1}{2}$, we have $r_{k+1} \leq 1$. Substituting for r_{k+1} from Lemma 7.1, we get

$$-\frac{F_{k-1} - F_k r_1}{F_{k-2} - F_{k-1} r_1} \leq 1.$$

Multiplying the numerator and denominator by $(-1)^k$ yields

$$\frac{(-1)^{k+1}(F_{k-1} - F_k r_1)}{(-1)^k(F_{k-2} - F_{k-1} r_1)} \leq 1.$$

By Lemma 7.2, $(-1)^k(F_{k-2} - F_{k-1} r_1) > 0$, and therefore we can multiply both sides of the inequality above by $(-1)^k(F_{k-2} - F_{k-1} r_1)$ to obtain

$$(-1)^{k+1}(F_{k-1} - F_k r_1) \leq (-1)^k(F_{k-2} - F_{k-1} r_1).$$

Rearranging yields

$$(-1)^{k+1}(F_{k-1} + F_k)r_1 \geq (-1)^{k+1}(F_{k-2} + F_{k-1}).$$

Using the law of formation of the Fibonacci sequence, we get

$$(-1)^{k+1}F_{k+1}r_1 \geq (-1)^{k+1}F_k,$$

which upon dividing by F_{k+1} on both sides gives the desired result. ■

We are now ready to prove the optimality of the Fibonacci search method and the uniqueness of this optimal solution.

Theorem 7.1 *Let r_1, \dots, r_N , $N \geq 2$, satisfy the constraints*

$$\begin{aligned} r_{k+1} &= \frac{1}{r_k} - 1, \quad k = 1, \dots, N-1, \\ r_k &\geq \frac{1}{2}, \quad k = 1, \dots, N. \end{aligned}$$

Then,

$$r_1 \cdots r_N \geq \frac{1}{F_{N+1}}.$$

Furthermore,

$$r_1 \cdots r_N = \frac{1}{F_{N+1}}$$

if and only if $r_k = F_{N-k+1}/F_{N-k+2}$, $k = 1, \dots, N$. In other words, the values of r_1, \dots, r_N used in the Fibonacci search method form a unique solution to the optimization problem. □

Proof. By substituting expressions for r_1, \dots, r_N from Lemma 7.1 and performing the appropriate cancellations, we obtain

$$r_1 \cdots r_N = (-1)^N (F_{N-2} - F_{N-1}r_1) = (-1)^N F_{N-2} + F_{N-1}(-1)^{N+1}r_1.$$

Using Lemma 7.3 yields

$$\begin{aligned} r_1 \cdots r_N &\geq (-1)^N F_{N-2} + F_{N-1}(-1)^{N+1} \frac{F_N}{F_{N+1}} \\ &= (-1)^N (F_{N-2}F_{N+1} - F_{N-1}F_N) \frac{1}{F_{N+1}}. \end{aligned}$$

By Exercise 7.5, it is readily checked that the following identity holds: $(-1)^N (F_{N-2}F_{N+1} - F_{N-1}F_N) = 1$. Hence,

$$r_1 \cdots r_N \geq \frac{1}{F_{N+1}}.$$

From the above we see that

$$r_1 \cdots r_N = \frac{1}{F_{N+1}}$$

if and only if

$$r_1 = \frac{F_N}{F_{N+1}}.$$

This is simply the value of r_1 for the Fibonacci search method. Note that fixing r_1 determines r_2, \dots, r_N uniquely. ■

For further discussion on the Fibonacci search method and its variants, see [133].

7.4 Bisection Method

Again we consider finding the minimizer of an objective function $f : \mathbb{R} \rightarrow \mathbb{R}$ over an interval $[a_0, b_0]$. As before, we assume that the objective function f is unimodal. Further, suppose that f is continuously differentiable and that we can use values of the derivative f' as a basis for reducing the uncertainty interval.

The *bisection method* is a simple algorithm for successively reducing the uncertainty interval based on evaluations of the derivative. To begin, let $x^{(0)} = (a_0 + b_0)/2$ be the midpoint of the initial uncertainty interval. Next, evaluate $f'(x^{(0)})$. If $f'(x^{(0)}) > 0$, then we deduce that the minimizer lies to the *left* of $x^{(0)}$. In other words, we reduce the uncertainty interval to $[a_0, x^{(0)}]$. On the other hand, if $f'(x^{(0)}) < 0$, then we deduce that the minimizer lies to the *right* of $x^{(0)}$. In this case, we reduce the uncertainty interval to $[x^{(0)}, b_0]$. Finally, if $f'(x^{(0)}) = 0$, then we declare $x^{(0)}$ to be the minimizer and terminate our search.

With the new uncertainty interval computed, we repeat the process iteratively. At each iteration k , we compute the midpoint of the uncertainty interval. Call this point $x^{(k)}$. Depending on the sign of $f'(x^{(k)})$ (assuming that it is nonzero), we reduce the uncertainty interval to the left or right of $x^{(k)}$. If at any iteration k we find that $f'(x^{(k)}) = 0$, then we declare $x^{(k)}$ to be the minimizer and terminate our search.

Two salient features distinguish the bisection method from the golden section and Fibonacci methods. First, instead of using values of f , the bisection methods uses values of f' . Second, at each iteration, the length of the uncertainty interval is reduced by a factor of $1/2$. Hence, after N steps, the range is reduced by a factor of $(1/2)^N$. This factor is smaller than in the golden section and Fibonacci methods.

Example 7.3 Recall Example 7.1 where we wish to find the minimizer of

$$f(x) = x^4 - 14x^3 + 60x^2 - 70x$$

in the interval $[0, 2]$ to within a range of 0.3. The golden section method requires at least four stages of reduction. If, instead, we use the bisection method, we would choose N so that

$$(0.5)^N \leq 0.3/2.$$

In this case, only three stages of reduction are needed. ■

7.5 Newton's Method

Suppose again that we are confronted with the problem of minimizing a function f of a single real variable x . We assume now that at each measurement

point $x^{(k)}$ we can determine $f(x^{(k)})$, $f'(x^{(k)})$, and $f''(x^{(k)})$. We can fit a quadratic function through $x^{(k)}$ that matches its first and second derivatives with that of the function f . This quadratic has the form

$$q(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}) + \frac{1}{2}f''(x^{(k)})(x - x^{(k)})^2.$$

Note that $q(x^{(k)}) = f(x^{(k)})$, $q'(x^{(k)}) = f'(x^{(k)})$, and $q''(x^{(k)}) = f''(x^{(k)})$. Then, instead of minimizing f , we minimize its approximation q . The first-order necessary condition for a minimizer of q yields

$$0 = q'(x) = f'(x^{(k)}) + f''(x^{(k)})(x - x^{(k)}).$$

Setting $x = x^{(k+1)}$, we obtain

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.$$

Example 7.4 Using Newton's method, we will find the minimizer of

$$f(x) = \frac{1}{2}x^2 - \sin x.$$

Suppose that the initial value is $x^{(0)} = 0.5$, and that the required accuracy is $\epsilon = 10^{-5}$, in the sense that we stop when $|x^{(k+1)} - x^{(k)}| < \epsilon$.

We compute

$$f'(x) = x - \cos x, \quad f''(x) = 1 + \sin x.$$

Hence,

$$\begin{aligned} x^{(1)} &= 0.5 - \frac{0.5 - \cos 0.5}{1 + \sin 0.5} \\ &= 0.5 - \frac{-0.3775}{1.479} \\ &= 0.7552. \end{aligned}$$

Proceeding in a similar manner, we obtain

$$\begin{aligned} x^{(2)} &= x^{(1)} - \frac{f'(x^{(1)})}{f''(x^{(1)})} = x^{(1)} - \frac{0.02710}{1.685} = 0.7391, \\ x^{(3)} &= x^{(2)} - \frac{f'(x^{(2)})}{f''(x^{(2)})} = x^{(2)} - \frac{9.461 \times 10^{-5}}{1.673} = 0.7390, \\ x^{(4)} &= x^{(3)} - \frac{f'(x^{(3)})}{f''(x^{(3)})} = x^{(3)} - \frac{1.17 \times 10^{-9}}{1.673} = 0.7390. \end{aligned}$$

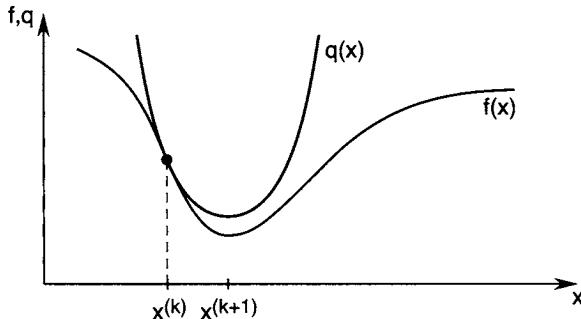


Figure 7.6 Newton's algorithm with $f''(x) > 0$.

Note that $|x^{(4)} - x^{(3)}| < \epsilon = 10^{-5}$. Furthermore, $f'(x^{(4)}) = -8.6 \times 10^{-6} \approx 0$. Observe that $f''(x^{(4)}) = 1.673 > 0$, so we can assume that $x^* \approx x^{(4)}$ is a strict minimizer. ■

Newton's method works well if $f''(x) > 0$ everywhere (see Figure 7.6). However, if $f''(x) < 0$ for some x , Newton's method may fail to converge to the minimizer (see Figure 7.7).

Newton's method can also be viewed as a way to drive the first derivative of f to zero. Indeed, if we set $g(x) = f'(x)$, then we obtain a formula for iterative solution of the equation $g(x) = 0$:

$$x^{(k+1)} = x^{(k)} - \frac{g(x^{(k)})}{g'(x^{(k)})}.$$

In other words, we can use Newton's method for zero finding.

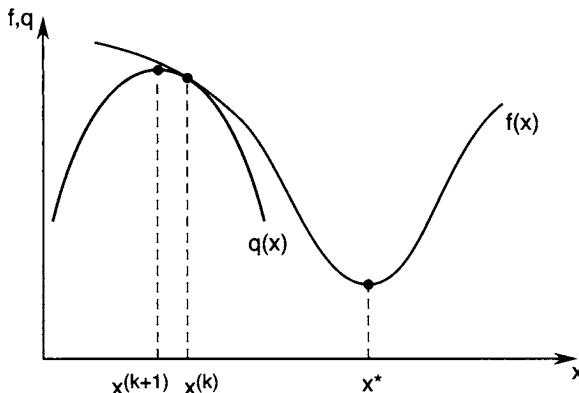


Figure 7.7 Newton's algorithm with $f''(x) < 0$.

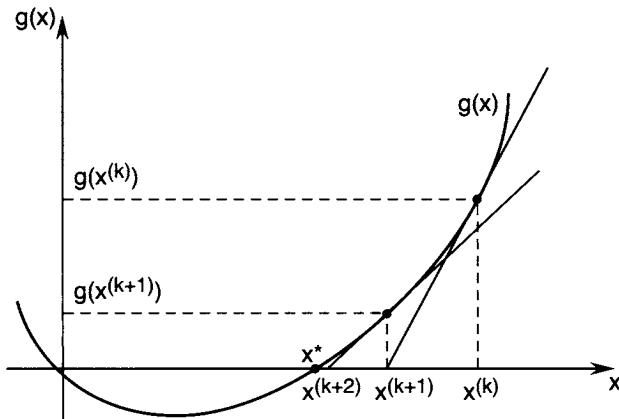


Figure 7.8 Newton's method of tangents.

Example 7.5 We apply Newton's method to improve a first approximation, $x^{(0)} = 12$, to the root of the equation

$$g(x) = x^3 - 12.2x^2 + 7.45x + 42 = 0.$$

We have $g'(x) = 3x^2 - 24.4x + 7.45$.

Performing two iterations yields

$$x^{(1)} = 12 - \frac{102.6}{146.65} = 11.33,$$

$$x^{(2)} = 11.33 - \frac{14.73}{116.11} = 11.21.$$

■

Newton's method for solving equations of the form $g(x) = 0$ is also referred to as *Newton's method of tangents*. This name is easily justified if we look at a geometric interpretation of the method when applied to the solution of the equation $g(x) = 0$ (see Figure 7.8).

If we draw a tangent to $g(x)$ at the given point $x^{(k)}$, then the tangent line intersects the x -axis at the point $x^{(k+1)}$, which we expect to be closer to the root x^* of $g(x) = 0$. Note that the slope of $g(x)$ at $x^{(k)}$ is

$$g'(x^{(k)}) = \frac{g(x^{(k)})}{x^{(k)} - x^{(k+1)}}.$$

Hence,

$$x^{(k+1)} = x^{(k)} - \frac{g(x^{(k)})}{g'(x^{(k)})}.$$

Newton's method of tangents may fail if the first approximation to the root is such that the ratio $g(x^{(0)})/g'(x^{(0)})$ is not small enough (see Figure 7.9). Thus, an initial approximation to the root is very important.

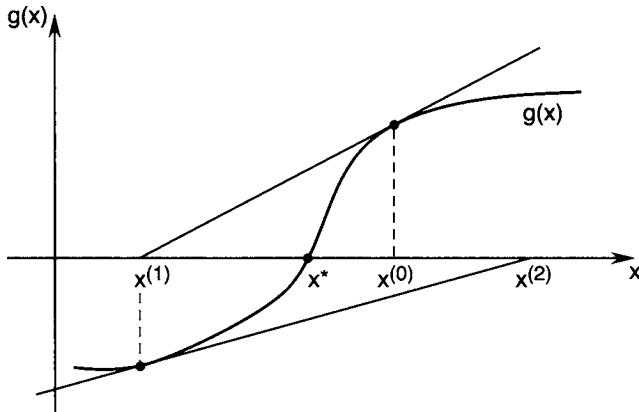


Figure 7.9 Example where Newton’s method of tangents fails to converge to the root x^* of $g(x) = 0$.

7.6 Secant Method

Newton’s method for minimizing f uses second derivatives of f :

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.$$

If the second derivative is not available, we may attempt to approximate it using first derivative information. In particular, we may approximate $f''(x^{(k)})$ above with

$$\frac{f'(x^{(k)}) - f'(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}.$$

Using the foregoing approximation of the second derivative, we obtain the algorithm

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{f'(x^{(k)}) - f'(x^{(k-1)})} f'(x^{(k)}),$$

called the *secant method*. Note that the algorithm requires two initial points to start it, which we denote $x^{(-1)}$ and $x^{(0)}$. The secant algorithm can be represented in the following equivalent form:

$$x^{(k+1)} = \frac{f'(x^{(k)})x^{(k-1)} - f'(x^{(k-1)})x^{(k)}}{f'(x^{(k)}) - f'(x^{(k-1)})}.$$

Observe that, like Newton’s method, the secant method does not directly involve values of $f(x^{(k)})$. Instead, it tries to drive the derivative f' to zero. In fact, as we did for Newton’s method, we can interpret the secant method as an algorithm for solving equations of the form $g(x) = 0$. Specifically, the

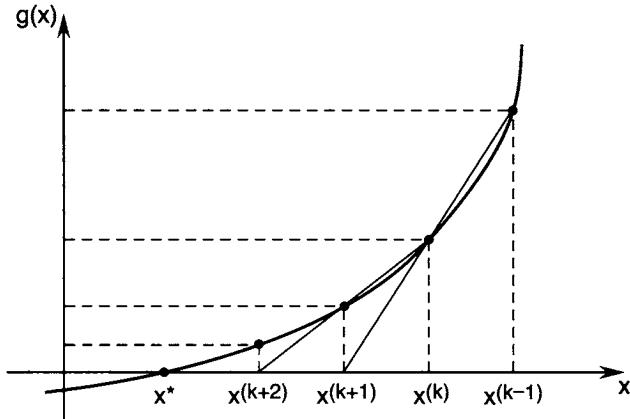


Figure 7.10 Secant method for root finding.

secant algorithm for finding a root of the equation $g(x) = 0$ takes the form

$$x^{(k+1)} = x^{(k)} - \frac{x^{(k)} - x^{(k-1)}}{g(x^{(k)}) - g(x^{(k-1)})} g(x^{(k)}),$$

or, equivalently,

$$x^{(k+1)} = \frac{g(x^{(k)})x^{(k-1)} - g(x^{(k-1)})x^{(k)}}{g(x^{(k)}) - g(x^{(k-1)})}.$$

The secant method for root finding is illustrated in Figure 7.10 (compare this with Figure 7.8). Unlike Newton's method, which uses the slope of g to determine the next point, the secant method uses the “secant” between the $(k - 1)$ th and k th points to determine the $(k + 1)$ th point.

Example 7.6 We apply the secant method to find the root of the equation

$$g(x) = x^3 - 12.2x^2 + 7.45x + 42 = 0.$$

We perform two iterations, with starting points $x^{(-1)} = 13$ and $x^{(0)} = 12$. We obtain

$$\begin{aligned} x^{(1)} &= 11.40, \\ x^{(2)} &= 11.25. \end{aligned}$$

■

Example 7.7 Suppose that the voltage across a resistor in a circuit decays according to the model $V(t) = e^{-Rt}$, where $V(t)$ is the voltage at time t and R is the resistance value.

Given measurements V_1, \dots, V_n of the voltage at times t_1, \dots, t_n , respectively, we wish to find the best estimate of R . By the *best estimate* we mean the value of R that minimizes the total squared error between the measured voltages and the voltages predicted by the model.

We derive an algorithm to find the best estimate of R using the secant method. The objective function is

$$f(R) = \sum_{i=1}^n (V_i - e^{-Rt_i})^2.$$

Hence, we have

$$f'(R) = 2 \sum_{i=1}^n (V_i - e^{-Rt_i}) e^{-Rt_i} t_i.$$

The secant algorithm for the problem is

$$\begin{aligned} R_{k+1} &= R_k - \frac{R_k - R_{k-1}}{\sum_{i=1}^n (V_i - e^{-R_k t_i}) e^{-R_k t_i} t_i - (V_i - e^{-R_{k-1} t_i}) e^{-R_{k-1} t_i} t_i} \\ &\quad \times \sum_{i=1}^n (V_i - e^{-R_k t_i}) e^{-R_k t_i} t_i. \end{aligned}$$

■

For further reading on the secant method, see [32]. Newton's method and the secant method are instances of *quadratic fit* methods. In Newton's method, $x^{(k+1)}$ is the stationary point of a quadratic function that fits f' and f'' at $x^{(k)}$. In the secant method, $x^{(k+1)}$ is the stationary point of a quadratic function that fits f' at $x^{(k)}$ and $x^{(k-1)}$. The secant method uses only f' (and not f'') but needs values from two previous points. We leave it to the reader to verify that if we set $x^{(k+1)}$ to be the stationary point of a quadratic function that fits f at $x^{(k)}$, $x^{(k-1)}$, and $x^{(k-2)}$, we obtain a quadratic fit method that uses only values of f :

$$x^{(k+1)} = \frac{\sigma_{12} f(x^{(k)}) + \sigma_{20} f(x^{(k-1)}) + \sigma_{01} f(x^{(k-2)})}{2(\delta_{12} f(x^{(k)}) + \delta_{20} f(x^{(k-1)}) + \delta_{01} f(x^{(k-2)}))}$$

where $\sigma_{ij} = (x^{(k-i)})^2 - (x^{(k-j)})^2$ and $\delta_{ij} = x^{(k-i)} - x^{(k-j)}$ (see Exercise 7.9). This method does not use f' or f'' , but needs values of f from three previous points. Three points are needed to initialize the iterations. The method is also sometimes called *inverse parabolic interpolation*.

An approach similar to fitting (or interpolation) based on higher-order polynomials is possible. For example, we could set $x^{(k+1)}$ to be a stationary point of a *cubic* function that fits f' at $x^{(k)}$, $x^{(k-1)}$, and $x^{(k-2)}$.

It is often practically advantageous to combine multiple methods, to overcome the limitations in any one method. For example, the golden section method is more robust but slower than inverse parabolic interpolation. *Brent's method* combines the two [17], resulting in a method that is faster than the golden section method but still retains its robustness properties.

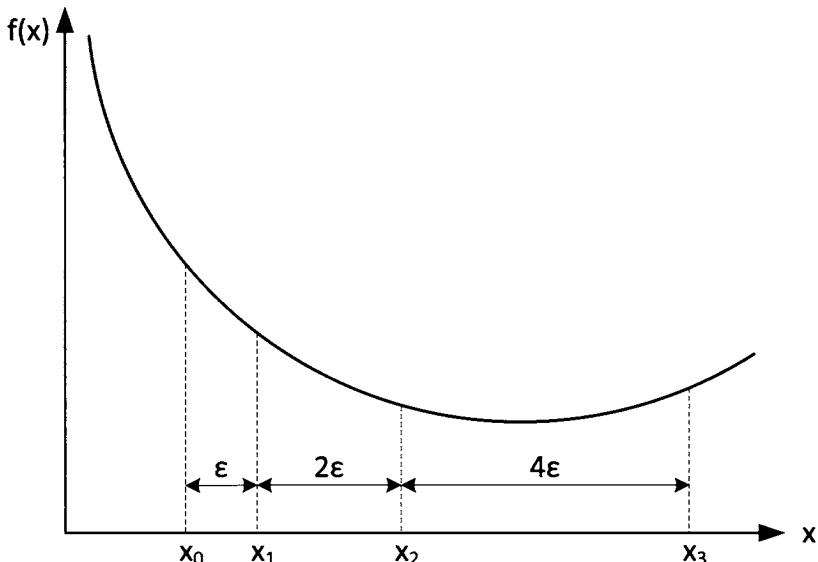


Figure 7.11 An illustration of the process of bracketing a minimizer.

7.7 Bracketing

Many of the methods we have described rely on an initial interval in which the minimizer is known to lie. This interval is also called a *bracket*, and procedures for finding such a bracket are called *bracketing* methods.

To find a bracket $[a, b]$ containing the minimizer, assuming unimodality, it suffices to find three points $a < c < b$ such that $f(c) < f(a)$ and $f(c) < f(b)$. A simple bracketing procedure is as follows. First, we pick three arbitrary points $x_0 < x_1 < x_2$. If $f(x_1) < f(x_0)$ and $f(x_1) < f(x_2)$, then we are done—the desired bracket is $[x_0, x_2]$. If not, say $f(x_0) > f(x_1) > f(x_2)$, then we pick a point $x_3 > x_2$ and check if $f(x_2) < f(x_3)$. If it holds, then again we are done—the desired bracket is $[x_1, x_3]$. Otherwise, we continue with this process until the function increases. Typically, each new point chosen involves an expansion in distance between successive test points. For example, we could double the distance between successive points, as illustrated in Figure 7.11. An analogous process applies if the initial three points are such that $f(x_0) < f(x_1) < f(x_2)$.

In the procedure described above, when the bracketing process terminates, we have three points x_{k-2} , x_{k-1} , and x_k such that $f(x_{k-1}) < f(x_{k-2})$ and $f(x_{k-1}) < f(x_k)$. The desired bracket is then $[x_{k-2}, x_k]$, which we can then use to initialize any of a number of search methods, including the golden section, Fibonacci, and bisection methods. Note that at this point, we have already evaluated $f(x_{k-2})$, $f(x_{k-1})$, and $f(x_k)$. If function evaluations are expensive to obtain, it would help if the point x_{k-1} inside the bracket also

coincides with one of the points used in the search method. For example, if we intend to use the golden section method, then it would help if $x_{k-1} - x_{k-2} = \rho(x_k - x_{k-2})$, where $\rho = (3 - \sqrt{5})/2$. In this case, x_{k-1} would be one of the two points within the initial interval used in the golden section method. This is achieved if each successive point x_k is chosen such that $x_k = x_{k-1} + (2 - \rho)(x_{k-1} - x_{k-2})$. In this case, the expansion in the distance between successive points is a factor $2 - \rho \approx 1.618$, which is less than double.

7.8 Line Search in Multidimensional Optimization

One-dimensional search methods play an important role in multidimensional optimization problems. In particular, iterative algorithms for solving such optimization problems (to be discussed in the following chapters) typically involve a *line search* at every iteration. To be specific, let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a function that we wish to minimize. Iterative algorithms for finding a minimizer of f are of the form

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},$$

where $\mathbf{x}^{(0)}$ is a given initial point and $\alpha_k \geq 0$ is chosen to minimize

$$\phi_k(\alpha) = f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}).$$

The vector $\mathbf{d}^{(k)}$ is called the *search direction* and α_k is called the *step size*. Figure 7.12 illustrates a line search within a multidimensional setting. Note that choice of α_k involves a one-dimensional minimization. This choice ensures that under appropriate conditions,

$$f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)}).$$

Any of the one-dimensional methods discussed in this chapter (including bracketing) can be used to minimize ϕ_k . We may, for example, use the secant method to find α_k . In this case we need the derivative of ϕ_k , which is

$$\phi'_k(\alpha) = \mathbf{d}^{(k)\top} \nabla f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}).$$

This is obtained using the chain rule. Therefore, applying the secant method for the line search requires the gradient ∇f , the initial line-search point $\mathbf{x}^{(k)}$, and the search direction $\mathbf{d}^{(k)}$ (see Exercise 7.11). Of course, other one-dimensional search methods may be used for line search (see, e.g., [43] and [88]).

Line-search algorithms used in practice involve considerations that we have not yet discussed thus far. First, determining the value of α_k that exactly minimizes ϕ_k may be computationally demanding; even worse, the minimizer of ϕ_k may not even exist. Second, practical experience suggests that it is better to allocate more computational time on iterating the multidimensional

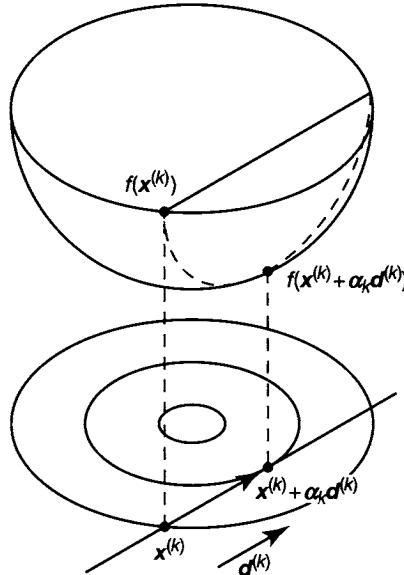


Figure 7.12 Line search in multidimensional optimization.

optimization algorithm rather than performing exact line searches. These considerations led to the development of conditions for terminating line-search algorithms that would result in low-accuracy line searches while still securing a sufficient decrease in the value of the f from one iteration to the next. The basic idea is that we have to ensure that the step size α_k is not too small or too large.

Some commonly used termination conditions are as follows. First, let $\varepsilon \in (0, 1)$, $\gamma > 1$, and $\eta \in (\varepsilon, 1)$ be given constants. The *Armijo condition* ensures that α_k is not too large by requiring that

$$\phi_k(\alpha_k) \leq \phi_k(0) + \varepsilon \alpha_k \phi'_k(0).$$

Further, it ensures that α_k is not too small by requiring that

$$\phi_k(\gamma \alpha_k) \geq \phi_k(0) + \varepsilon \gamma \alpha_k \phi'_k(0).$$

The *Goldstein condition* differs from Armijo in the second inequality:

$$\phi_k(\alpha_k) \geq \phi_k(0) + \eta \alpha_k \phi'_k(0).$$

The first Armijo inequality together with the Goldstein condition are often jointly called the *Armijo-Goldstein condition*. The *Wolfe condition* differs from Goldstein in that it involves only ϕ'_k :

$$\phi'_k(\alpha_k) \geq \eta \phi'_k(0).$$

A stronger variation of this is the *strong Wolfe condition*:

$$|\phi'_k(\alpha_k)| \leq \eta |\phi'_k(0)|.$$

A simple practical (inexact) line-search method is the *Armijo backtracking algorithm*, described as follows. We start with some candidate value for the step size α_k . If this candidate value satisfies a prespecified termination condition (usually the first Armijo inequality), then we stop and use it as the step size. Otherwise, we iteratively *decrease* the value by multiplying it by some constant factor $\tau \in (0, 1)$ (typically $\tau = 0.5$) and re-check the termination condition. If $\alpha^{(0)}$ is the initial candidate value, then after m iterations the value obtained is $\alpha_k = \tau^m \alpha^{(0)}$. The algorithm *backtracks* from the initial value until the termination condition holds. In other words, the algorithm produces a value for the step size of the form $\alpha_k = \tau^m \alpha^{(0)}$ with m being the smallest value in $\{0, 1, 2, \dots\}$ for which α_k satisfies the termination condition.

For more information on practical line-search methods, we refer the reader to [43, pp. 26–40], [96, Sec. 10.5], [11, App. C], [49], and [50].¹

EXERCISES

7.1 Suppose that we have a unimodal function over the interval $[5, 8]$. Give an example of a desired final uncertainty range where the golden section method requires at least four iterations, whereas the Fibonacci method requires only three. You may choose an arbitrarily small value of ε for the Fibonacci method.

7.2 Let $f(x) = x^2 + 4 \cos x$, $x \in \mathbb{R}$. We wish to find the minimizer x^* of f over the interval $[1, 2]$. (*Calculator users*: Note that in $\cos x$, the argument x is in radians.)

- a. Plot $f(x)$ versus x over the interval $[1, 2]$.
- b. Use the golden section method to locate x^* to within an uncertainty of 0.2. Display all intermediate steps using a table:

Iteration k	a_k	b_k	$f(a_k)$	$f(b_k)$	New uncertainty interval
1	?	?	?	?	[?, ?]
2	?	?	?	?	[?, ?]
⋮	⋮	⋮	⋮	⋮	⋮

- c. Repeat part b using the Fibonacci method, with $\varepsilon = 0.05$. Display all intermediate steps using a table:

¹We thank Dennis M. Goodman for furnishing us with references [49] and [50].

Iteration k	ρ_k	a_k	b_k	$f(a_k)$	$f(b_k)$	New uncertainty interval
1	?	?	?	?	?	[?,?]
2	?	?	?	?	?	[?,?]
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

- d. Apply Newton's method, using the same number of iterations as in part b, with $x^{(0)} = 1$.

7.3 Let $f(x) = 8e^{1-x} + 7\log(x)$, where “log” represents the natural logarithm function.

- a. Use MATLAB to plot $f(x)$ versus x over the interval $[1, 2]$, and verify that f is unimodal over $[1, 2]$.
- b. Write a simple MATLAB program to implement the golden section method that locates the minimizer of f over $[1, 2]$ to within an uncertainty of 0.23. Display all intermediate steps using a table as in Exercise 7.2.
- c. Repeat part b using the Fibonacci method, with $\varepsilon = 0.05$. Display all intermediate steps using a table as in Exercise 7.2.

7.4 Suppose that ρ_1, \dots, ρ_N are the values used in the Fibonacci search method. Show that for each $k = 1, \dots, N$, $0 \leq \rho_k \leq 1/2$, and for each $k = 1, \dots, N-1$,

$$\rho_{k+1} = 1 - \frac{\rho_k}{1 - \rho_k}.$$

7.5 Show that if F_0, F_1, \dots is the Fibonacci sequence, then for each $k = 2, 3, \dots$,

$$F_{k-2}F_{k+1} - F_{k-1}F_k = (-1)^k.$$

7.6 Show that the Fibonacci sequence can be calculated using the formula

$$F_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^{n+1} - \left(\frac{1-\sqrt{5}}{2} \right)^{n+1} \right).$$

7.7 Suppose that we have an efficient way of calculating exponentials. Based on this, use Newton's method to devise a method to approximate $\log(2)$ [where “log” is the natural logarithm function]. Use an initial point of $x^{(0)} = 1$, and perform two iterations.

7.8 Consider the problem of finding the zero of $g(x) = (e^x - 1)/(e^x + 1)$, $x \in \mathbb{R}$, where e^x is the exponential of x . (Note that 0 is the unique zero of g .)

- a. Write down the algorithm for Newton's method of tangents applied to this problem. Simplify using the identity $\sinh x = (e^x - e^{-x})/2$.
- b. Find an initial condition $x^{(0)}$ such that the algorithm cycles [i.e., $x^{(0)} = x^{(2)} = x^{(4)} = \dots$]. You need not explicitly calculate the initial condition; it suffices to provide an equation that the initial condition must satisfy.
Hint: Draw a graph of g .
- c. For what values of the initial condition does the algorithm converge?

7.9 Derive a one-dimensional search (minimization) algorithm based on quadratic fit with only objective function values. Specifically, derive an algorithm that computes $x^{(k+1)}$ based on $x^{(k)}$, $x^{(k-1)}$, $x^{(k-2)}$, $f(x^{(k)})$, $f(x^{(k-1)})$, and $f(x^{(k-2)})$.

Hint: To simplify, use the notation $\sigma_{ij} = (x^{(k-i)})^2 - (x^{(k-j)})^2$ and $\delta_{ij} = x^{(k-i)} - x^{(k-j)}$. You might also find it useful to experiment with your algorithm by writing a MATLAB program. Note that three points are needed to initialize the algorithm.

7.10 The objective of this exercise is to implement the secant method using MATLAB.

- a. Write a simple MATLAB program to implement the secant method to locate the root of the equation $g(x) = 0$. For the stopping criterion, use the condition $|x^{(k+1)} - x^{(k)}| < |x^{(k)}|\varepsilon$, where $\varepsilon > 0$ is a given constant.
- b. Let $g(x) = (2x - 1)^2 + 4(4 - 1024x)^4$. Find the root of $g(x) = 0$ using the secant method with $x^{(-1)} = 0$, $x^{(0)} = 1$, and $\varepsilon = 10^{-5}$. Also determine the value of g at the solution obtained.

7.11 Write a MATLAB function that implements a line search algorithm using the secant method. The arguments to this function are the name of the M-file for the gradient, the current point, and the search direction. For example, the function may be called `linesearch_secant` and be used by the function call `alpha=linesearch_secant('grad',x,d)`, where `grad.m` is the M-file containing the gradient, `x` is the starting line search point, `d` is the search direction, and `alpha` is the value returned by the function [which we use in the following chapters as the step size for iterative algorithms (see, e.g., Exercises 8.25 and 10.11)].

Note: In the solutions manual, we used the stopping criterion $|\mathbf{d}^\top \nabla f(\mathbf{x} + \alpha\mathbf{d})| \leq \varepsilon |\mathbf{d}^\top \nabla f(\mathbf{x})|$, where $\varepsilon > 0$ is a prespecified number, ∇f is the gradient, \mathbf{x} is the starting line search point, and \mathbf{d} is the search direction. The rationale for the stopping criterion above is that we want to reduce the directional derivative of f in the direction \mathbf{d} by the specified fraction ε . We used a value of $\varepsilon = 10^{-4}$ and initial conditions of 0 and 0.001.

7.12 Consider using a gradient algorithm to minimize the function

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{x}$$

with the initial guess $\mathbf{x}^{(0)} = [0.8, -0.25]^\top$.

- a. To initialize the line search, apply the bracketing procedure in Figure 7.11 along the line starting at $\mathbf{x}^{(0)}$ in the direction of the negative gradient. Use $\varepsilon = 0.075$.
- b. Apply the golden section method to reduce the width of the uncertainty region to 0.01. Organize the results of your computation in a table format similar to that of Exercise 7.2.
- c. Repeat the above using the Fibonacci method.

CHAPTER 8

GRADIENT METHODS

8.1 Introduction

In this chapter we consider a class of search methods for real-valued functions on \mathbb{R}^n . These methods use the gradient of the given function. In our discussion we use such terms as *level sets*, *normal vectors*, and *tangent vectors*. These notions were discussed in some detail in Part I.

Recall that a level set of a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is the set of points \mathbf{x} satisfying $f(\mathbf{x}) = c$ for some constant c . Thus, a point $\mathbf{x}_0 \in \mathbb{R}^n$ is on the level set corresponding to level c if $f(\mathbf{x}_0) = c$. In the case of functions of two real variables, $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, the notion of the level set is illustrated in Figure 8.1.

The gradient of f at \mathbf{x}_0 , denoted $\nabla f(\mathbf{x}_0)$, if it is not a zero vector, is orthogonal to the tangent vector to an arbitrary smooth curve passing through \mathbf{x}_0 on the level set $f(\mathbf{x}) = c$. Thus, the direction of maximum rate of increase of a real-valued differentiable function at a point is orthogonal to the level set of the function through that point. In other words, the gradient acts in such a direction that for a given small displacement, the function f increases more in the direction of the gradient than in any other direction. To prove this statement, recall that $\langle \nabla f(\mathbf{x}), \mathbf{d} \rangle, \|\mathbf{d}\| = 1$, is the rate of increase of f in

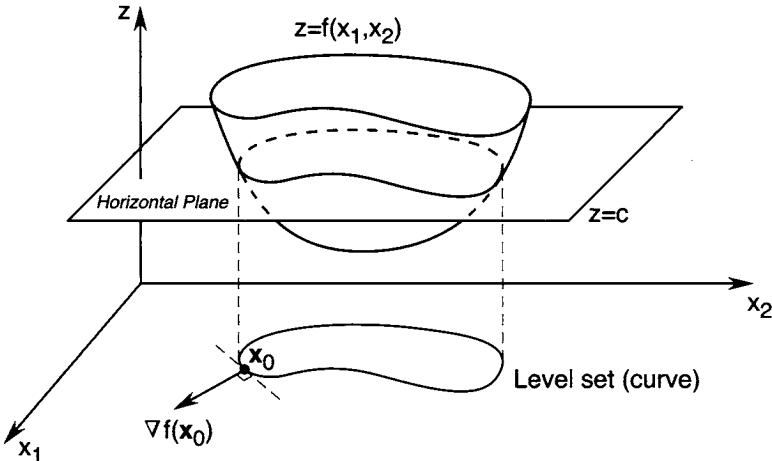


Figure 8.1 Constructing a level set corresponding to level c for f .

the direction \mathbf{d} at the point \mathbf{x} . By the Cauchy-Schwarz inequality,

$$\langle \nabla f(\mathbf{x}), \mathbf{d} \rangle \leq \|\nabla f(\mathbf{x})\|$$

because $\|\mathbf{d}\| = 1$. But if $\mathbf{d} = \nabla f(\mathbf{x})/\|\nabla f(\mathbf{x})\|$, then

$$\left\langle \nabla f(\mathbf{x}), \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|} \right\rangle = \|\nabla f(\mathbf{x})\|.$$

Thus, the direction in which $\nabla f(\mathbf{x})$ points is the direction of maximum rate of increase of f at \mathbf{x} . The direction in which $-\nabla f(\mathbf{x})$ points is the direction of maximum rate of decrease of f at \mathbf{x} . Hence, the direction of negative gradient is a good direction to search if we want to find a function minimizer.

We proceed as follows. Let $\mathbf{x}^{(0)}$ be a starting point, and consider the point $\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)})$. Then, by Taylor's theorem, we obtain

$$f(\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)})) = f(\mathbf{x}^{(0)}) - \alpha \|\nabla f(\mathbf{x}^{(0)})\|^2 + o(\alpha).$$

Thus, if $\nabla f(\mathbf{x}^{(0)}) \neq \mathbf{0}$, then for sufficiently small $\alpha > 0$, we have

$$f(\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)})) < f(\mathbf{x}^{(0)}).$$

This means that the point $\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)})$ is an improvement over the point $\mathbf{x}^{(0)}$ if we are searching for a minimizer.

To formulate an algorithm that implements this idea, suppose that we are given a point $\mathbf{x}^{(k)}$. To find the next point $\mathbf{x}^{(k+1)}$, we start at $\mathbf{x}^{(k)}$ and move by an amount $-\alpha_k \nabla f(\mathbf{x}^{(k)})$, where α_k is a positive scalar called the *step size*. This procedure leads to the following iterative algorithm:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)}).$$

We refer to this as a *gradient descent algorithm* (or simply a *gradient algorithm*). The gradient varies as the search proceeds, tending to zero as we approach the minimizer. We have the option of either taking very small steps and reevaluating the gradient at every step, or we can take large steps each time. The first approach results in a laborious method of reaching the minimizer, whereas the second approach may result in a more zigzag path to the minimizer. The advantage of the second approach is possibly fewer gradient evaluations. Among many different methods that use this philosophy the most popular is the method of *steepest descent*, which we discuss next.

Gradient methods are simple to implement and often perform well. For this reason, they are used widely in practical applications. For a discussion of applications of the steepest descent method to the computation of optimal controllers, we recommend [85, pp. 481–515]. In Chapter 13 we apply a gradient method to the training of a class of neural networks.

8.2 The Method of Steepest Descent

The method of steepest descent is a gradient algorithm where the step size α_k is chosen to achieve the maximum amount of decrease of the objective function at each individual step. Specifically, α_k is chosen to minimize $\phi_k(\alpha) \triangleq f(\mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)}))$. In other words,

$$\alpha_k = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)})).$$

To summarize, the steepest descent algorithm proceeds as follows: At each step, starting from the point $\mathbf{x}^{(k)}$, we conduct a line search in the direction $-\nabla f(\mathbf{x}^{(k)})$ until a minimizer, $\mathbf{x}^{(k+1)}$, is found. A typical sequence resulting from the method of steepest descent is depicted in Figure 8.2.

Observe that the method of steepest descent moves in orthogonal steps, as stated in the following proposition.

Proposition 8.1 *If $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ is a steepest descent sequence for a given function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, then for each k the vector $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ is orthogonal to the vector $\mathbf{x}^{(k+2)} - \mathbf{x}^{(k+1)}$.* \square

Proof. From the iterative formula of the method of steepest descent it follows that

$$\langle \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}, \mathbf{x}^{(k+2)} - \mathbf{x}^{(k+1)} \rangle = \alpha_k \alpha_{k+1} \langle \nabla f(\mathbf{x}^{(k)}), \nabla f(\mathbf{x}^{(k+1)}) \rangle.$$

To complete the proof it is enough to show that

$$\langle \nabla f(\mathbf{x}^{(k)}), \nabla f(\mathbf{x}^{(k+1)}) \rangle = 0.$$

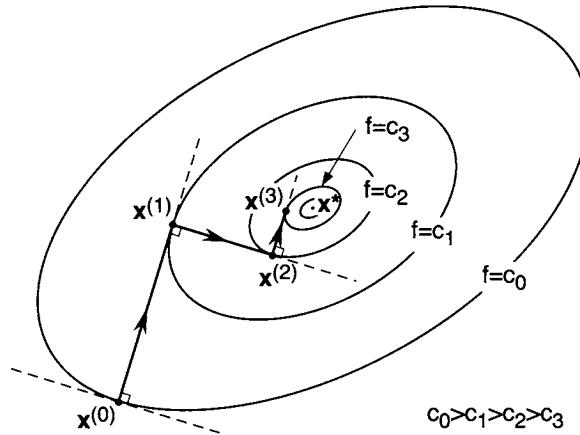


Figure 8.2 Typical sequence resulting from the method of steepest descent.

To this end, observe that α_k is a nonnegative scalar that minimizes $\phi_k(\alpha) \triangleq f(\mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)}))$. Hence, using the FONC and the chain rule gives us

$$\begin{aligned} 0 &= \phi'_k(\alpha_k) \\ &= \frac{d\phi_k}{d\alpha}(\alpha_k) \\ &= \nabla f(\mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)}))^T (-\nabla f(\mathbf{x}^{(k)})) \\ &= -\langle \nabla f(\mathbf{x}^{(k+1)}), \nabla f(\mathbf{x}^{(k)}) \rangle, \end{aligned}$$

which completes the proof. ■

The proposition above implies that $\nabla f(\mathbf{x}^{(k)})$ is parallel to the tangent plane to the level set $\{f(\mathbf{x}) = f(\mathbf{x}^{(k+1)})\}$ at $\mathbf{x}^{(k+1)}$. Note that as each new point is generated by the steepest descent algorithm, the corresponding value of the function f decreases in value, as stated below.

Proposition 8.2 *If $\{\mathbf{x}^{(k)}\}_{k=0}^{\infty}$ is the steepest descent sequence for $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and if $\nabla f(\mathbf{x}^{(k)}) \neq \mathbf{0}$, then $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$.* □

Proof. First recall that

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)}),$$

where $\alpha_k \geq 0$ is the minimizer of

$$\phi_k(\alpha) = f(\mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)}))$$

over all $\alpha \geq 0$. Thus, for $\alpha \geq 0$, we have

$$\phi_k(\alpha_k) \leq \phi_k(\alpha).$$

By the chain rule,

$$\phi'_k(0) = \frac{d\phi_k}{d\alpha}(0) = -(\nabla f(\mathbf{x}^{(k)}) - 0\nabla f(\mathbf{x}^{(k)}))^\top \nabla f(\mathbf{x}^{(k)}) = -\|\nabla f(\mathbf{x}^{(k)})\|^2 < 0$$

because $\nabla f(\mathbf{x}^{(k)}) \neq \mathbf{0}$ by assumption. Thus, $\phi'_k(0) < 0$ and this implies that there is an $\bar{\alpha} > 0$ such that $\phi_k(0) > \phi_k(\alpha)$ for all $\alpha \in (0, \bar{\alpha}]$. Hence,

$$f(\mathbf{x}^{(k+1)}) = \phi_k(\alpha_k) \leq \phi_k(\bar{\alpha}) < \phi_k(0) = f(\mathbf{x}^{(k)}),$$

which completes the proof. ■

In Proposition 8.2, we proved that the algorithm possesses the *descent property*: $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$ if $\nabla f(\mathbf{x}^{(k)}) \neq \mathbf{0}$. If for some k , we have $\nabla f(\mathbf{x}^{(k)}) = \mathbf{0}$, then the point $\mathbf{x}^{(k)}$ satisfies the FONC. In this case, $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$. We can use the above as the basis for a stopping (termination) criterion for the algorithm.

The condition $\nabla f(\mathbf{x}^{(k+1)}) = \mathbf{0}$, however, is not directly suitable as a practical stopping criterion, because the numerical computation of the gradient will rarely be identically equal to zero. A practical stopping criterion is to check if the norm $\|\nabla f(\mathbf{x}^{(k)})\|$ of the gradient is less than a prespecified threshold, in which case we stop. Alternatively, we may compute the absolute difference $|f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})|$ between objective function values for every two successive iterations, and if the difference is less than some prespecified threshold, then we stop; that is, we stop when

$$|f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})| < \varepsilon,$$

where $\varepsilon > 0$ is a prespecified threshold. Yet another alternative is to compute the norm $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|$ of the difference between two successive iterates, and we stop if the norm is less than a prespecified threshold:

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| < \varepsilon.$$

Alternatively, we may check “relative” values of the quantities above; for example,

$$\frac{|f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})|}{|f(\mathbf{x}^{(k)})|} < \varepsilon$$

or

$$\frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|}{\|\mathbf{x}^{(k)}\|} < \varepsilon.$$

The two (relative) stopping criteria above are preferable to the previous (absolute) criteria because the relative criteria are “scale-independent.” For example, scaling the objective function does not change the satisfaction of the criterion $|f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})|/|f(\mathbf{x}^{(k)})| < \varepsilon$. Similarly, scaling the decision variable does not change the satisfaction of the criterion $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|/\|\mathbf{x}^{(k)}\| < \varepsilon$.

To avoid dividing by very small numbers, we can modify these stopping criteria as follows:

$$\frac{|f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})|}{\max\{1, |f(\mathbf{x}^{(k)})|\}} < \varepsilon$$

or

$$\frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|}{\max\{1, \|\mathbf{x}^{(k)}\|\}} < \varepsilon.$$

Note that the stopping criteria above are relevant to all the iterative algorithms we discuss in this part.

Example 8.1 We use the method of steepest descent to find the minimizer of

$$f(x_1, x_2, x_3) = (x_1 - 4)^4 + (x_2 - 3)^2 + 4(x_3 + 5)^4.$$

The initial point is $\mathbf{x}^{(0)} = [4, 2, -1]^\top$. We perform three iterations.

We find that

$$\nabla f(\mathbf{x}) = [4(x_1 - 4)^3, 2(x_2 - 3), 16(x_3 + 5)^3]^\top.$$

Hence,

$$\nabla f(\mathbf{x}^{(0)}) = [0, -2, 1024]^\top.$$

To compute $\mathbf{x}^{(1)}$, we need

$$\begin{aligned}\alpha_0 &= \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)})) \\ &= \arg \min_{\alpha \geq 0} (0 + (2 + 2\alpha - 3)^2 + 4(-1 - 1024\alpha + 5)^4) \\ &= \arg \min_{\alpha \geq 0} \phi_0(\alpha).\end{aligned}$$

Using the secant method from Section 7.6, we obtain

$$\alpha_0 = 3.967 \times 10^{-3}.$$

For illustrative purpose, we show a plot of $\phi_0(\alpha)$ versus α in Figure 8.3, obtained using MATLAB. Thus,

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \alpha_0 \nabla f(\mathbf{x}^{(0)}) = [4.000, 2.008, -5.062]^\top.$$

To find $\mathbf{x}^{(2)}$, we first determine

$$\nabla f(\mathbf{x}^{(1)}) = [0.000, -1.984, -0.003875]^\top.$$

Next, we find α_1 , where

$$\begin{aligned}\alpha_1 &= \arg \min_{\alpha \geq 0} (0 + (2.008 + 1.984\alpha - 3)^2 + 4(-5.062 + 0.003875\alpha + 5)^4) \\ &= \arg \min_{\alpha \geq 0} \phi_1(\alpha).\end{aligned}$$

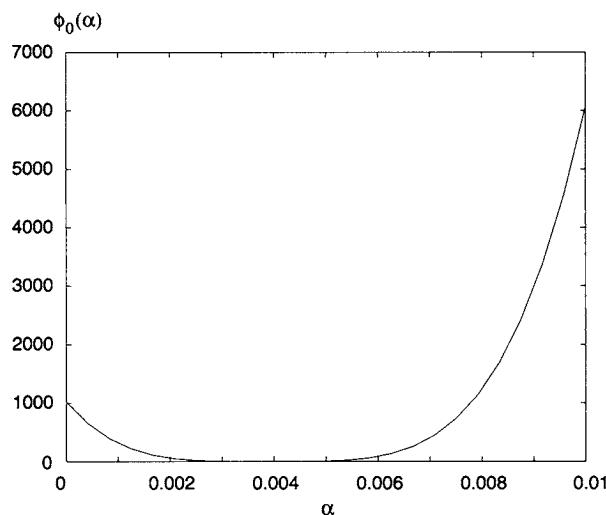


Figure 8.3 Plot of $\phi_0(\alpha)$ versus α .

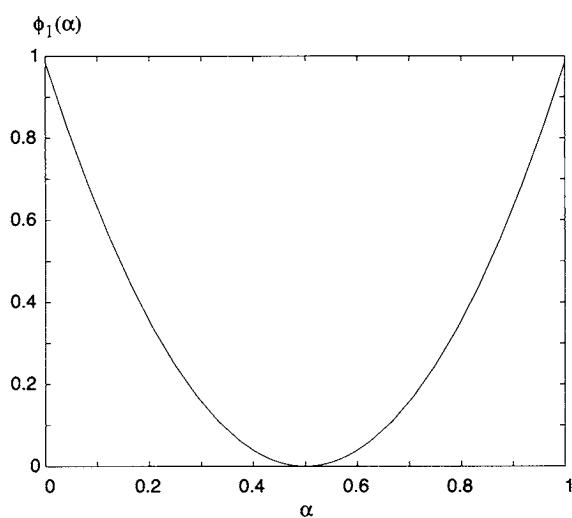


Figure 8.4 Plot of $\phi_1(\alpha)$ versus α .

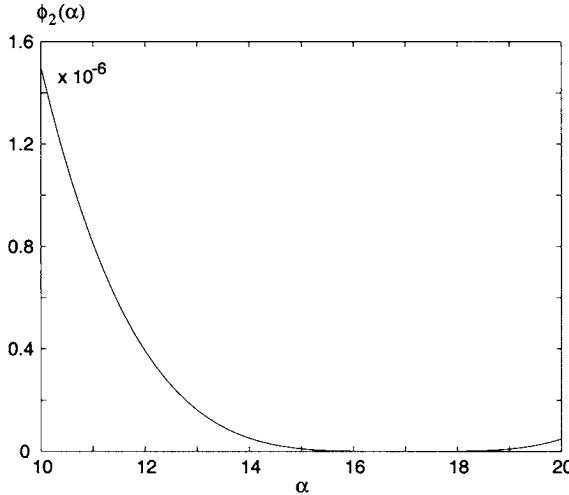


Figure 8.5 Plot of $\phi_2(\alpha)$ versus α .

Using the secant method again, we obtain $\alpha_1 = 0.5000$. Figure 8.4 depicts a plot of $\phi_1(\alpha)$ versus α . Thus,

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} - \alpha_1 \nabla f(\mathbf{x}^{(1)}) = [4.000, 3.000, -5.060]^\top.$$

To find $\mathbf{x}^{(3)}$, we first determine

$$\nabla f(\mathbf{x}^{(2)}) = [0.000, 0.000, -0.003525]^\top$$

and

$$\begin{aligned}\alpha_2 &= \arg \min_{\alpha \geq 0} (0.000 + 0.000 + 4(-5.060 + 0.003525\alpha + 5)^4) \\ &= \arg \min_{\alpha \geq 0} \phi_2(\alpha).\end{aligned}$$

We proceed as in the previous iterations to obtain $\alpha_2 = 16.29$. A plot of $\phi_2(\alpha)$ versus α is shown in Figure 8.5.

The value of $\mathbf{x}^{(3)}$ is

$$\mathbf{x}^{(3)} = [4.000, 3.000, -5.002]^\top.$$

Note that the minimizer of f is $[4, 3, -5]^\top$, and hence it appears that we have arrived at the minimizer in only three iterations. The reader should be cautioned not to draw any conclusions from this example about the number of iterations required to arrive at a solution in general.

It goes without saying that numerical computations, such as those in this example, are performed in practice using a computer (rather than by hand).

The calculations above were written out explicitly, step by step, for the purpose of illustrating the operations involved in the steepest descent algorithm. The computations themselves were, in fact, carried out using a MATLAB program (see Exercise 8.25). ■

Let us now see what the method of steepest descent does with a quadratic function of the form

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} - \mathbf{b}^\top \mathbf{x},$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix, $\mathbf{b} \in \mathbb{R}^n$, and $\mathbf{x} \in \mathbb{R}^n$. The unique minimizer of f can be found by setting the gradient of f to zero, where

$$\nabla f(\mathbf{x}) = \mathbf{Q}\mathbf{x} - \mathbf{b},$$

because $D(\mathbf{x}^\top \mathbf{Q} \mathbf{x}) = \mathbf{x}^\top (\mathbf{Q} + \mathbf{Q}^\top) = 2\mathbf{x}^\top \mathbf{Q}$, and $D(\mathbf{b}^\top \mathbf{x}) = \mathbf{b}^\top$. There is no loss of generality in assuming \mathbf{Q} to be a symmetric matrix. For if we are given a quadratic form $\mathbf{x}^\top \mathbf{A} \mathbf{x}$ and $\mathbf{A} \neq \mathbf{A}^\top$, then because the transposition of a scalar equals itself, we obtain

$$(\mathbf{x}^\top \mathbf{A} \mathbf{x})^\top = \mathbf{x}^\top \mathbf{A}^\top \mathbf{x} = \mathbf{x}^\top \mathbf{A} \mathbf{x}.$$

Hence,

$$\begin{aligned} \mathbf{x}^\top \mathbf{A} \mathbf{x} &= \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} + \frac{1}{2} \mathbf{x}^\top \mathbf{A}^\top \mathbf{x} \\ &= \frac{1}{2} \mathbf{x}^\top (\mathbf{A} + \mathbf{A}^\top) \mathbf{x} \\ &\triangleq \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x}. \end{aligned}$$

Note that

$$(\mathbf{A} + \mathbf{A}^\top)^\top = \mathbf{Q}^\top = \mathbf{A} + \mathbf{A}^\top = \mathbf{Q}.$$

The Hessian of f is $\mathbf{F}(\mathbf{x}) = \mathbf{Q} = \mathbf{Q}^\top > 0$. To simplify the notation we write $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$. Then, the steepest descent algorithm for the quadratic function can be represented as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{g}^{(k)},$$

where

$$\begin{aligned} \alpha_k &= \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} - \alpha \mathbf{g}^{(k)}) \\ &= \arg \min_{\alpha \geq 0} \left(\frac{1}{2} (\mathbf{x}^{(k)} - \alpha \mathbf{g}^{(k)})^\top \mathbf{Q} (\mathbf{x}^{(k)} - \alpha \mathbf{g}^{(k)}) - (\mathbf{x}^{(k)} - \alpha \mathbf{g}^{(k)})^\top \mathbf{b} \right). \end{aligned}$$

In the quadratic case, we can find an explicit formula for α_k . We proceed as follows. Assume that $\mathbf{g}^{(k)} \neq \mathbf{0}$, for if $\mathbf{g}^{(k)} = \mathbf{0}$, then $\mathbf{x}^{(k)} = \mathbf{x}^*$ and the

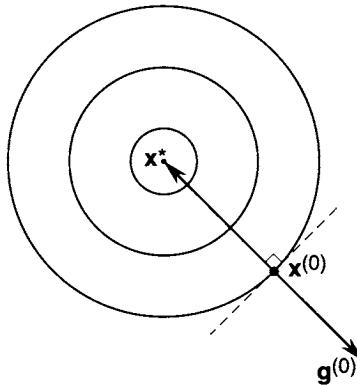


Figure 8.6 Steepest descent method applied to $f(x_1, x_2) = x_1^2 + x_2^2$.

algorithm stops. Because $\alpha_k \geq 0$ is a minimizer of $\phi_k(\alpha) = f(\mathbf{x}^{(k)} - \alpha \mathbf{g}^{(k)})$, we apply the FONC to $\phi_k(\alpha)$ to obtain

$$\phi'_k(\alpha) = (\mathbf{x}^{(k)} - \alpha \mathbf{g}^{(k)})^\top \mathbf{Q}(-\mathbf{g}^{(k)}) - \mathbf{b}^\top(-\mathbf{g}^{(k)}).$$

Therefore, $\phi'_k(\alpha) = 0$ if $\alpha \mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)} = (\mathbf{x}^{(k)\top} \mathbf{Q} - \mathbf{b}^\top) \mathbf{g}^{(k)}$. But

$$\mathbf{x}^{(k)\top} \mathbf{Q} - \mathbf{b}^\top = \mathbf{g}^{(k)\top}.$$

Hence,

$$\alpha_k = \frac{\mathbf{g}^{(k)\top} \mathbf{g}^{(k)}}{\mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)}}.$$

In summary, the method of steepest descent for the quadratic takes the form

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \frac{\mathbf{g}^{(k)\top} \mathbf{g}^{(k)}}{\mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)}} \mathbf{g}^{(k)},$$

where

$$\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)}) = \mathbf{Q} \mathbf{x}^{(k)} - \mathbf{b}.$$

Example 8.2 Let

$$f(x_1, x_2) = x_1^2 + x_2^2.$$

Then, starting from an arbitrary initial point $\mathbf{x}^{(0)} \in \mathbb{R}^2$, we arrive at the solution $\mathbf{x}^* = \mathbf{0} \in \mathbb{R}^2$ in only one step. See Figure 8.6.

However, if

$$f(x_1, x_2) = \frac{x_1^2}{5} + x_2^2,$$

then the method of steepest descent shuffles ineffectively back and forth when searching for the minimizer in a narrow valley (see Figure 8.7). This example illustrates a major drawback in the steepest descent method. More

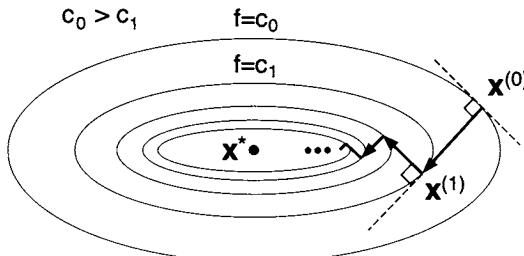


Figure 8.7 Steepest descent method in search for minimizer in a narrow valley.

sophisticated methods that alleviate this problem are discussed in subsequent chapters. ■

To understand better the method of steepest descent, we examine its convergence properties in the next section.

8.3 Analysis of Gradient Methods

Convergence

The method of steepest descent is an example of an iterative algorithm. This means that the algorithm generates a sequence of points, each calculated on the basis of the points preceding it. The method is a *descent method* because as each new point is generated by the algorithm, the corresponding value of the objective function decreases in value (i.e., the algorithm possesses the descent property).

We say that an iterative algorithm is *globally convergent* if for any arbitrary starting point the algorithm is guaranteed to generate a sequence of points converging to a point that satisfies the FONC for a minimizer. When the algorithm is not globally convergent, it may still generate a sequence that converges to a point satisfying the FONC, provided that the initial point is sufficiently close to the point. In this case we say that the algorithm is *locally convergent*. How close to a solution point we need to start for the algorithm to converge depends on the local convergence properties of the algorithm. A related issue of interest pertaining to a given locally or globally convergent algorithm is the *rate of convergence*; that is, how fast the algorithm converges to a solution point.

In this section we analyze the convergence properties of descent gradient methods, including the method of steepest descent and gradient methods with fixed step size. We can investigate important convergence characteristics of a gradient method by applying the method to quadratic problems. The convergence analysis is more convenient if instead of working with f we deal

with

$$V(\mathbf{x}) = f(\mathbf{x}) + \frac{1}{2} \mathbf{x}^{*\top} \mathbf{Q} \mathbf{x}^* = \frac{1}{2} (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{Q} (\mathbf{x} - \mathbf{x}^*),$$

where $\mathbf{Q} = \mathbf{Q}^\top > 0$. The solution point \mathbf{x}^* is obtained by solving $\mathbf{Q}\mathbf{x} = \mathbf{b}$; that is, $\mathbf{x}^* = \mathbf{Q}^{-1}\mathbf{b}$. The function V differs from f only by a constant $\frac{1}{2}\mathbf{x}^{*\top} \mathbf{Q} \mathbf{x}^*$. We begin our analysis with the following useful lemma that applies to a general gradient algorithm.

Lemma 8.1 *The iterative algorithm*

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{g}^{(k)}$$

with $\mathbf{g}^{(k)} = \mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b}$ satisfies

$$V(\mathbf{x}^{(k+1)}) = (1 - \gamma_k) V(\mathbf{x}^{(k)}),$$

where if $\mathbf{g}^{(k)} = \mathbf{0}$, then $\gamma_k = 1$, and if $\mathbf{g}^{(k)} \neq \mathbf{0}$, then

$$\gamma_k = \alpha_k \frac{\mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)}}{\mathbf{g}^{(k)\top} \mathbf{Q}^{-1} \mathbf{g}^{(k)}} \left(2 \frac{\mathbf{g}^{(k)\top} \mathbf{g}^{(k)}}{\mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)}} - \alpha_k \right).$$

□

Proof. The proof is by direct computation. Note that if $\mathbf{g}^{(k)} = \mathbf{0}$, then the desired result holds trivially. In the remainder of the proof, assume that $\mathbf{g}^{(k)} \neq \mathbf{0}$. We first evaluate the expression

$$\frac{V(\mathbf{x}^{(k)}) - V(\mathbf{x}^{(k+1)})}{V(\mathbf{x}^{(k)})}.$$

To facilitate computations, let $\mathbf{y}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^*$. Then, $V(\mathbf{x}^{(k)}) = \frac{1}{2} \mathbf{y}^{(k)\top} \mathbf{Q} \mathbf{y}^{(k)}$. Hence,

$$\begin{aligned} V(\mathbf{x}^{(k+1)}) &= \frac{1}{2} (\mathbf{x}^{(k+1)} - \mathbf{x}^*)^\top \mathbf{Q} (\mathbf{x}^{(k+1)} - \mathbf{x}^*) \\ &= \frac{1}{2} (\mathbf{x}^{(k)} - \mathbf{x}^* - \alpha_k \mathbf{g}^{(k)})^\top \mathbf{Q} (\mathbf{x}^{(k)} - \mathbf{x}^* - \alpha_k \mathbf{g}^{(k)}) \\ &= \frac{1}{2} \mathbf{y}^{(k)\top} \mathbf{Q} \mathbf{y}^{(k)} - \alpha_k \mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{y}^{(k)} + \frac{1}{2} \alpha_k^2 \mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)}. \end{aligned}$$

Therefore,

$$\frac{V(\mathbf{x}^{(k)}) - V(\mathbf{x}^{(k+1)})}{V(\mathbf{x}^{(k)})} = \frac{2\alpha_k \mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{y}^{(k)} - \alpha_k^2 \mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)}}{\mathbf{y}^{(k)\top} \mathbf{Q} \mathbf{y}^{(k)}}.$$

Because

$$\mathbf{g}^{(k)} = \mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b} = \mathbf{Q}\mathbf{x}^{(k)} - \mathbf{Q}\mathbf{x}^* = \mathbf{Q}\mathbf{y}^{(k)},$$

we have

$$\begin{aligned}\mathbf{y}^{(k)\top} \mathbf{Q} \mathbf{y}^{(k)} &= \mathbf{g}^{(k)\top} \mathbf{Q}^{-1} \mathbf{g}^{(k)}, \\ \mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{y}^{(k)} &= \mathbf{g}^{(k)\top} \mathbf{g}^{(k)}.\end{aligned}$$

Therefore, substituting the above, we get

$$\frac{V(\mathbf{x}^{(k)}) - V(\mathbf{x}^{(k+1)})}{V(\mathbf{x}^{(k)})} = \alpha_k \frac{\mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)}}{\mathbf{g}^{(k)\top} \mathbf{Q}^{-1} \mathbf{g}^{(k)}} \left(2 \frac{\mathbf{g}^{(k)\top} \mathbf{g}^{(k)}}{\mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)}} - \alpha_k \right) = \gamma_k.$$

■

Note that $\gamma_k \leq 1$ for all k , because $\gamma_k = 1 - V(\mathbf{x}^{(k+1)})/V(\mathbf{x}^{(k)})$ and V is a nonnegative function. If $\gamma_k = 1$ for some k , then $V(\mathbf{x}^{(k+1)}) = 0$, which is equivalent to $\mathbf{x}^{(k+1)} = \mathbf{x}^*$. In this case we also have that for all $i \geq k+1$, $\mathbf{x}^{(i)} = \mathbf{x}^*$ and $\gamma_i = 1$. It turns out that $\gamma_k = 1$ if and only if either $\mathbf{g}^{(k)} = \mathbf{0}$ or $\mathbf{g}^{(k)}$ is an eigenvector of \mathbf{Q} (see Lemma 8.3).

We are now ready to state and prove our key convergence theorem for gradient methods. The theorem gives a necessary and sufficient condition for the sequence $\{\mathbf{x}^{(k)}\}$ generated by a gradient method to converge to \mathbf{x}^* ; that is, $\mathbf{x}^{(k)} \rightarrow \mathbf{x}^*$ or $\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \mathbf{x}^*$.

Theorem 8.1 *Let $\{\mathbf{x}^{(k)}\}$ be the sequence resulting from a gradient algorithm $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{g}^{(k)}$. Let γ_k be as defined in Lemma 8.1, and suppose that $\gamma_k > 0$ for all k . Then, $\{\mathbf{x}^{(k)}\}$ converges to \mathbf{x}^* for any initial condition $\mathbf{x}^{(0)}$ if and only if*

$$\sum_{k=0}^{\infty} \gamma_k = \infty.$$

□

Proof. From Lemma 8.1 we have $V(\mathbf{x}^{(k+1)}) = (1 - \gamma_k) V(\mathbf{x}^{(k)})$, from which we obtain

$$V(\mathbf{x}^{(k)}) = \left(\prod_{i=0}^{k-1} (1 - \gamma_i) \right) V(\mathbf{x}^{(0)}).$$

Assume that $\gamma_k < 1$ for all k , for otherwise the result holds trivially. Note that $\mathbf{x}^{(k)} \rightarrow \mathbf{x}^*$ if and only if $V(\mathbf{x}^{(k)}) \rightarrow 0$. By the equation above we see that this occurs if and only if $\prod_{i=0}^{\infty} (1 - \gamma_i) = 0$, which, in turn, holds if and only if $\sum_{i=0}^{\infty} -\log(1 - \gamma_i) = \infty$ (we get this simply by taking logs). Note that by Lemma 8.1, $1 - \gamma_i \geq 0$ and $\log(1 - \gamma_i)$ is well-defined [$\log(0)$ is taken to be $-\infty$]. Therefore, it remains to show that $\sum_{i=0}^{\infty} -\log(1 - \gamma_i) = \infty$ if and only if

$$\sum_{i=0}^{\infty} \gamma_i = \infty.$$

We first show that $\sum_{i=0}^{\infty} \gamma_i = \infty$ implies that $\sum_{i=0}^{\infty} -\log(1 - \gamma_i) = \infty$. For this, first observe that for any $x \in \mathbb{R}$, $x > 0$, we have $\log(x) \leq x - 1$

[this is easy to see simply by plotting $\log(x)$ and $x - 1$ versus x]. Therefore, $\log(1 - \gamma_i) \leq 1 - \gamma_i - 1 = -\gamma_i$, and hence $-\log(1 - \gamma_i) \geq \gamma_i$. Thus, if $\sum_{i=0}^{\infty} \gamma_i = \infty$, then clearly $\sum_{i=0}^{\infty} -\log(1 - \gamma_i) = \infty$.

Finally, we show that $\sum_{i=0}^{\infty} -\log(1 - \gamma_i) = \infty$ implies that $\sum_{i=0}^{\infty} \gamma_i = \infty$. We proceed by contraposition. Suppose that $\sum_{i=0}^{\infty} \gamma_i < \infty$. Then, it must be that $\gamma_i \rightarrow 0$. Now observe that for $x \in \mathbb{R}$, $x \leq 1$ and x sufficiently close to 1, we have $\log(x) \geq 2(x - 1)$ [as before, this is easy to see simply by plotting $\log(x)$ and $2(x - 1)$ versus x]. Therefore, for sufficiently large i , $\log(1 - \gamma_i) \geq 2(1 - \gamma_i - 1) = -2\gamma_i$, which implies that $-\log(1 - \gamma_i) \leq 2\gamma_i$. Hence, $\sum_{i=0}^{\infty} \gamma_i < \infty$ implies that $\sum_{i=0}^{\infty} -\log(1 - \gamma_i) < \infty$.

This completes the proof. ■

The assumption in Theorem 8.1 that $\gamma_k > 0$ for all k is significant in that it corresponds to the algorithm having the descent property (see Exercise 8.23). Furthermore, the result of the theorem does not hold in general if we do not assume that $\gamma_k > 0$ for all k , as shown in the following example.

Example 8.3 We show, using a counterexample, that the assumption that $\gamma_k > 0$ in Theorem 8.1 is necessary for the result of the theorem to hold. Indeed, for each $k = 0, 1, 2, \dots$, choose α_k in such a way that $\gamma_{2k} = -1/2$ and $\gamma_{2k+1} = 1/2$ (we can always do this if, for example, $\mathbf{Q} = \mathbf{I}_n$). From Lemma 8.1 we have

$$V(\mathbf{x}^{(2(k+1))}) = (1 - 1/2)(1 + 1/2)V(\mathbf{x}^{(2k)}) = (3/4)V(\mathbf{x}^{(2k)}).$$

Therefore, $V(\mathbf{x}^{(2k)}) \rightarrow 0$. Because $V(\mathbf{x}^{(2k+1)}) = (3/2)V(\mathbf{x}^{(2k)})$, we also have that $V(\mathbf{x}^{(2k+1)}) \rightarrow 0$. Hence, $V(\mathbf{x}^{(k)}) \rightarrow 0$, which implies that $\mathbf{x}^{(k)} \rightarrow 0$ (for all $\mathbf{x}^{(0)}$). On the other hand, it is clear that

$$\sum_{i=0}^k \gamma_i \leq \frac{1}{2}$$

for all k . Hence, the result of the theorem does not hold if $\gamma_k \leq 0$ for some k . ■

Using the general theorem above, we can now establish the convergence of specific cases of the gradient algorithm, including the steepest descent algorithm and algorithms with fixed step size. In the analysis to follow, we use Rayleigh's inequality, which states that for any $\mathbf{Q} = \mathbf{Q}^\top > 0$, we have

$$\lambda_{\min}(\mathbf{Q})\|\mathbf{x}\|^2 \leq \mathbf{x}^\top \mathbf{Q} \mathbf{x} \leq \lambda_{\max}(\mathbf{Q})\|\mathbf{x}\|^2,$$

where $\lambda_{\min}(\mathbf{Q})$ denotes the minimal eigenvalue of \mathbf{Q} and $\lambda_{\max}(\mathbf{Q})$ denotes the maximal eigenvalue of \mathbf{Q} . For $\mathbf{Q} = \mathbf{Q}^\top > 0$, we also have

$$\begin{aligned} \lambda_{\min}(\mathbf{Q}^{-1}) &= \frac{1}{\lambda_{\max}(\mathbf{Q})}, \\ \lambda_{\max}(\mathbf{Q}^{-1}) &= \frac{1}{\lambda_{\min}(\mathbf{Q})}, \end{aligned}$$

and

$$\lambda_{\min}(\mathbf{Q}^{-1})\|\mathbf{x}\|^2 \leq \mathbf{x}^\top \mathbf{Q}^{-1} \mathbf{x} \leq \lambda_{\max}(\mathbf{Q}^{-1})\|\mathbf{x}\|^2.$$

Lemma 8.2 Let $\mathbf{Q} = \mathbf{Q}^\top > 0$ be an $n \times n$ real symmetric positive definite matrix. Then, for any $\mathbf{x} \in \mathbb{R}^n$, we have

$$\frac{\lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbf{Q})} \leq \frac{(\mathbf{x}^\top \mathbf{x})^2}{(\mathbf{x}^\top \mathbf{Q} \mathbf{x})(\mathbf{x}^\top \mathbf{Q}^{-1} \mathbf{x})} \leq \frac{\lambda_{\max}(\mathbf{Q})}{\lambda_{\min}(\mathbf{Q})}.$$

□

Proof. Applying Rayleigh's inequality and using the properties of symmetric positive definite matrices listed previously, we get

$$\frac{(\mathbf{x}^\top \mathbf{x})^2}{(\mathbf{x}^\top \mathbf{Q} \mathbf{x})(\mathbf{x}^\top \mathbf{Q}^{-1} \mathbf{x})} \leq \frac{\|\mathbf{x}\|^4}{\lambda_{\min}(\mathbf{Q})\|\mathbf{x}\|^2\lambda_{\min}(\mathbf{Q}^{-1})\|\mathbf{x}\|^2} = \frac{\lambda_{\max}(\mathbf{Q})}{\lambda_{\min}(\mathbf{Q})}$$

and

$$\frac{(\mathbf{x}^\top \mathbf{x})^2}{(\mathbf{x}^\top \mathbf{Q} \mathbf{x})(\mathbf{x}^\top \mathbf{Q}^{-1} \mathbf{x})} \geq \frac{\|\mathbf{x}\|^4}{\lambda_{\max}(\mathbf{Q})\|\mathbf{x}\|^2\lambda_{\max}(\mathbf{Q}^{-1})\|\mathbf{x}\|^2} = \frac{\lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbf{Q})}.$$

■

We are now ready to establish the convergence of the steepest descent method.

Theorem 8.2 In the steepest descent algorithm, we have $\mathbf{x}^{(k)} \rightarrow \mathbf{x}^*$ for any $\mathbf{x}^{(0)}$. □

Proof. If $\mathbf{g}^{(k)} = \mathbf{0}$ for some k , then $\mathbf{x}^{(k)} = \mathbf{x}^*$ and the result holds. So assume that $\mathbf{g}^{(k)} \neq \mathbf{0}$ for all k . Recall that for the steepest descent algorithm,

$$\alpha_k = \frac{\mathbf{g}^{(k)\top} \mathbf{g}^{(k)}}{\mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)}}.$$

Substituting this expression for α_k in the formula for γ_k yields

$$\gamma_k = \frac{(\mathbf{g}^{(k)\top} \mathbf{g}^{(k)})^2}{(\mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)})(\mathbf{g}^{(k)\top} \mathbf{Q}^{-1} \mathbf{g}^{(k)})}.$$

Note that in this case $\gamma_k > 0$ for all k . Furthermore, by Lemma 8.2, we have $\gamma_k \geq (\lambda_{\min}(\mathbf{Q})/\lambda_{\max}(\mathbf{Q})) > 0$. Therefore, we have $\sum_{k=0}^{\infty} \gamma_k = \infty$, and hence by Theorem 8.1 we conclude that $\mathbf{x}^{(k)} \rightarrow \mathbf{x}^*$. ■

Consider now a gradient method with fixed step size; that is, $\alpha_k = \alpha \in \mathbb{R}$ for all k . The resulting algorithm is of the form

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \mathbf{g}^{(k)}.$$

We refer to the algorithm above as a *fixed-step-size* gradient algorithm. The algorithm is of practical interest because of its simplicity. In particular, the algorithm does not require a line search at each step to determine α_k , because the same step size α is used at each step. Clearly, the convergence of the algorithm depends on the choice of α , and we would not expect the algorithm to work for arbitrary α . The following theorem gives a necessary and sufficient condition on α for convergence of the algorithm.

Theorem 8.3 *For the fixed-step-size gradient algorithm, $\mathbf{x}^{(k)} \rightarrow \mathbf{x}^*$ for any $\mathbf{x}^{(0)}$ if and only if*

$$0 < \alpha < \frac{2}{\lambda_{\max}(\mathbf{Q})}.$$

□

Proof. \Leftarrow : By Rayleigh's inequality we have

$$\lambda_{\min}(\mathbf{Q})\mathbf{g}^{(k)\top}\mathbf{g}^{(k)} \leq \mathbf{g}^{(k)\top}\mathbf{Q}\mathbf{g}^{(k)} \leq \lambda_{\max}(\mathbf{Q})\mathbf{g}^{(k)\top}\mathbf{g}^{(k)}$$

and

$$\mathbf{g}^{(k)\top}\mathbf{Q}^{-1}\mathbf{g}^{(k)} \leq \frac{1}{\lambda_{\min}(\mathbf{Q})}\mathbf{g}^{(k)\top}\mathbf{g}^{(k)}.$$

Therefore, substituting the above into the formula for γ_k , we get

$$\gamma_k \geq \alpha (\lambda_{\min}(\mathbf{Q}))^2 \left(\frac{2}{\lambda_{\max}(\mathbf{Q})} - \alpha \right) > 0.$$

Therefore, $\gamma_k > 0$ for all k , and $\sum_{k=0}^{\infty} \gamma_k = \infty$. Hence, by Theorem 8.1 we conclude that $\mathbf{x}^{(k)} \rightarrow \mathbf{x}^*$.

\Rightarrow : We use contraposition. Suppose that either $\alpha \leq 0$ or $\alpha \geq 2/\lambda_{\max}(\mathbf{Q})$. Let $\mathbf{x}^{(0)}$ be chosen such that $\mathbf{x}^{(0)} - \mathbf{x}^*$ is an eigenvector of \mathbf{Q} corresponding to the eigenvalue $\lambda_{\max}(\mathbf{Q})$. Because

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha(\mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b}) = \mathbf{x}^{(k)} - \alpha(\mathbf{Q}\mathbf{x}^{(k)} - \mathbf{Q}\mathbf{x}^*),$$

we obtain

$$\begin{aligned} \mathbf{x}^{(k+1)} - \mathbf{x}^* &= \mathbf{x}^{(k)} - \mathbf{x}^* - \alpha(\mathbf{Q}\mathbf{x}^{(k)} - \mathbf{Q}\mathbf{x}^*) \\ &= (\mathbf{I}_n - \alpha\mathbf{Q})(\mathbf{x}^{(k)} - \mathbf{x}^*) \\ &= (\mathbf{I}_n - \alpha\mathbf{Q})^{k+1}(\mathbf{x}^{(0)} - \mathbf{x}^*) \\ &= (1 - \alpha\lambda_{\max}(\mathbf{Q}))^{k+1}(\mathbf{x}^{(0)} - \mathbf{x}^*), \end{aligned}$$

where in the last line we used the property that $\mathbf{x}^{(0)} - \mathbf{x}^*$ is an eigenvector of \mathbf{Q} . Taking norms on both sides, we get

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| = |1 - \alpha\lambda_{\max}(\mathbf{Q})|^{k+1} \|\mathbf{x}^{(0)} - \mathbf{x}^*\|.$$

Because $\alpha \leq 0$ or $\alpha \geq 2/\lambda_{\max}(\mathbf{Q})$,

$$|1 - \alpha\lambda_{\max}(\mathbf{Q})| \geq 1.$$

Hence, $\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|$ cannot converge to 0, and thus the sequence $\{\mathbf{x}^{(k)}\}$ does not converge to \mathbf{x}^* . ■

Example 8.4 Let the function f be given by

$$f(\mathbf{x}) = \mathbf{x}^\top \begin{bmatrix} 4 & 2\sqrt{2} \\ 0 & 5 \end{bmatrix} \mathbf{x} + \mathbf{x}^\top \begin{bmatrix} 3 \\ 6 \end{bmatrix} + 24.$$

We wish to find the minimizer of f using a fixed-step-size gradient algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)}),$$

where $\alpha \in \mathbb{R}$ is a fixed step size.

To apply Theorem 8.3, we first symmetrize the matrix in the quadratic term of f to get

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \begin{bmatrix} 8 & 2\sqrt{2} \\ 2\sqrt{2} & 10 \end{bmatrix} \mathbf{x} + \mathbf{x}^\top \begin{bmatrix} 3 \\ 6 \end{bmatrix} + 24.$$

The eigenvalues of the matrix in the quadratic term are 6 and 12. Hence, using Theorem 8.3, the algorithm converges to the minimizer for all $\mathbf{x}^{(0)}$ if and only if α lies in the range $0 < \alpha < 2/12$. ■

Convergence Rate

We now turn our attention to the issue of convergence rates of gradient algorithms. In particular, we focus on the steepest descent algorithm. We first present the following theorem.

Theorem 8.4 *In the method of steepest descent applied to the quadratic function, at every step k we have*

$$V(\mathbf{x}^{(k+1)}) \leq \frac{\lambda_{\max}(\mathbf{Q}) - \lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbf{Q})} V(\mathbf{x}^{(k)}).$$

□

Proof. In the proof of Theorem 8.2, we showed that $\gamma_k \geq \lambda_{\min}(\mathbf{Q})/\lambda_{\max}(\mathbf{Q})$. Therefore,

$$\frac{V(\mathbf{x}^{(k)}) - V(\mathbf{x}^{(k+1)})}{V(\mathbf{x}^{(k)})} = \gamma_k \geq \frac{\lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbf{Q})},$$

and the result follows. ■

Theorem 8.4 is relevant to our consideration of the convergence rate of the steepest descent algorithm as follows. Let

$$r = \frac{\lambda_{\max}(\mathbf{Q})}{\lambda_{\min}(\mathbf{Q})} = \|\mathbf{Q}\| \|\mathbf{Q}^{-1}\|,$$

called the *condition number* of \mathbf{Q} . Then, it follows from Theorem 8.4 that

$$V(\mathbf{x}^{(k+1)}) \leq \left(1 - \frac{1}{r}\right) V(\mathbf{x}^{(k)}).$$

The term $(1 - 1/r)$ plays an important role in the convergence of $\{V(\mathbf{x}^{(k)})\}$ to 0 (and hence of $\{\mathbf{x}^{(k)}\}$ to \mathbf{x}^*). We refer to $(1 - 1/r)$ as the *convergence ratio*. Specifically, we see that the smaller the value of $(1 - 1/r)$, the smaller $V(\mathbf{x}^{(k+1)})$ will be relative to $V(\mathbf{x}^{(k)})$, and hence the “faster” $V(\mathbf{x}^{(k)})$ converges to 0, as indicated by the inequality above. The convergence ratio $(1 - 1/r)$ decreases as r decreases. If $r = 1$, then $\lambda_{\max}(\mathbf{Q}) = \lambda_{\min}(\mathbf{Q})$, corresponding to circular contours of f (see Figure 8.6). In this case the algorithm converges in a single step to the minimizer. As r increases, the speed of convergence of $\{V(\mathbf{x}^{(k)})\}$ (and hence of $\{\mathbf{x}^{(k)}\}$) decreases. The increase in r reflects that fact that the contours of f are more eccentric (see, e.g., Figure 8.7). We refer the reader to [88, pp. 238, 239] for an alternative approach to the analysis above.

To investigate the convergence properties of $\{\mathbf{x}^{(k)}\}$ further, we need the following definition.

Definition 8.1 Given a sequence $\{\mathbf{x}^{(k)}\}$ that converges to \mathbf{x}^* , that is, $\lim_{k \rightarrow \infty} \|\mathbf{x}^{(k)} - \mathbf{x}^*\| = 0$, we say that the *order of convergence* is p , where $p \in \mathbb{R}$, if

$$0 < \lim_{k \rightarrow \infty} \frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|}{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^p} < \infty.$$

If for all $p > 0$,

$$\lim_{k \rightarrow \infty} \frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|}{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^p} = 0,$$

then we say that the order of convergence is ∞ . ■

Note that in the definition above, $0/0$ should be understood to be 0.

The order of convergence of a sequence is a measure of its rate of convergence; the higher the order, the faster the rate of convergence. The order of convergence is sometimes also called the *rate of convergence* (see, e.g., [96]). If $p = 1$ (first-order convergence) and $\lim_{k \rightarrow \infty} \|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| / \|\mathbf{x}^{(k)} - \mathbf{x}^*\| = 1$, we say that the convergence is *sublinear*. If $p = 1$ and $\lim_{k \rightarrow \infty} \|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| / \|\mathbf{x}^{(k)} - \mathbf{x}^*\| < 1$, we say that the convergence is *linear*. If $p > 1$, we say that the convergence is *superlinear*. If $p = 2$ (second-order convergence), we say that the convergence is *quadratic*.

Example 8.5 1. Suppose that $x^{(k)} = 1/k$ and thus $x^{(k)} \rightarrow 0$. Then,

$$\frac{|x^{(k+1)}|}{|x^{(k)}|^p} = \frac{1/(k+1)}{1/k^p} = \frac{k^p}{k+1}.$$

If $p < 1$, the sequence above converges to 0, whereas if $p > 1$, it grows to ∞ . If $p = 1$, the sequence converges to 1. Hence, the order of convergence is 1 (i.e., we have linear convergence).

2. Suppose that $x^{(k)} = \gamma^k$, where $0 < \gamma < 1$, and thus $x^{(k)} \rightarrow 0$. Then,

$$\frac{|x^{(k+1)}|}{|x^{(k)}|^p} = \frac{\gamma^{k+1}}{(\gamma^k)^p} = \gamma^{k+1-kp} = \gamma^{k(1-p)+1}.$$

If $p < 1$, the sequence above converges to 0, whereas if $p > 1$, it grows to ∞ . If $p = 1$, the sequence converges to γ (in fact, remains constant at γ). Hence, the order of convergence is 1.

3. Suppose that $x^{(k)} = \gamma^{(q^k)}$, where $q > 1$ and $0 < \gamma < 1$, and thus $x^{(k)} \rightarrow 0$. Then,

$$\frac{|x^{(k+1)}|}{|x^{(k)}|^p} = \frac{\gamma^{(q^{k+1})}}{(\gamma^{(q^k)})^p} = \gamma^{(q^{k+1}-pq^k)} = \gamma^{(q-p)q^k}.$$

If $p < q$, the sequence above converges to 0, whereas if $p > q$, it grows to ∞ . If $p = q$, the sequence converges to 1 (in fact, remains constant at 1). Hence, the order of convergence is q .

4. Suppose that $x^{(k)} = 1$ for all k , and thus $x^{(k)} \rightarrow 1$ trivially. Then,

$$\frac{|x^{(k+1)} - 1|}{|x^{(k)} - 1|^p} = \frac{0}{0^p} = 0$$

for all p . Hence, the order of convergence is ∞ . ■

The order of convergence can be interpreted using the notion of the order symbol O , as follows. Recall that $a = O(h)$ ("big-oh of h ") if there exists a constant c such that $|a| \leq c|h|$ for sufficiently small h . Then, the order of convergence is *at least* p if

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| = O(\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^p)$$

(see Theorem 8.5 below). For example, the order of convergence is at least 2 if

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| = O(\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2)$$

(this fact is used in the analysis of Newton's algorithm in Chapter 9).

Theorem 8.5 Let $\{\mathbf{x}^{(k)}\}$ be a sequence that converges to \mathbf{x}^* . If

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| = O(\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^p),$$

then the order of convergence (if it exists) is at least p . \square

Proof. Let s be the order of convergence of $\{\mathbf{x}^{(k)}\}$. Suppose that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| = O(\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^p).$$

Then, there exists c such that for sufficiently large k ,

$$\frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|}{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^p} \leq c.$$

Hence,

$$\begin{aligned} \frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|}{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^s} &= \frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|}{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^p} \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^{p-s} \\ &\leq c \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^{p-s}. \end{aligned}$$

Taking limits yields

$$\lim_{k \rightarrow \infty} \frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|}{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^s} \leq c \lim_{k \rightarrow \infty} \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^{p-s}.$$

Because by definition s is the order of convergence,

$$\lim_{k \rightarrow \infty} \frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|}{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^s} > 0.$$

Combining the two inequalities above, we get

$$c \lim_{k \rightarrow \infty} \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^{p-s} > 0.$$

Therefore, because $\lim_{k \rightarrow \infty} \|\mathbf{x}^{(k)} - \mathbf{x}^*\| = 0$, we conclude that $s \geq p$; that is, the order of convergence is at least p . \blacksquare

By an argument similar to the above, we can show that if

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| = o(\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^p),$$

then the order of convergence (if it exists) strictly exceeds p .

Example 8.6 Suppose that we are given a scalar sequence $\{x^{(k)}\}$ that converges with order of convergence p and satisfies

$$\lim_{k \rightarrow \infty} \frac{|x^{(k+1)} - 2|}{|x^{(k)} - 2|^3} = 0.$$

The limit of $\{x^{(k)}\}$ must be 2, because it is clear from the equation that $|x^{(k+1)} - 2| \rightarrow 0$. Also, we see that $|x^{(k+1)} - 2| = o(|x^{(k)} - 2|^3)$. Hence, we conclude that $p > 3$. ■

It turns out that the order of convergence of any convergent sequence cannot be less than 1 (see Exercise 8.3). In the following, we provide an example where the order of convergence of a fixed-step-size gradient algorithm exceeds 1.

Example 8.7 Consider the problem of finding a minimizer of the function $f : \mathbb{R} \rightarrow \mathbb{R}$ given by

$$f(x) = x^2 - \frac{x^3}{3}.$$

Suppose that we use the algorithm $x^{(k+1)} = x^{(k)} - \alpha f'(x^{(k)})$ with step size $\alpha = 1/2$ and initial condition $x^{(0)} = 1$. (The notation f' represents the derivative of f .)

We first show that the algorithm converges to a local minimizer of f . Indeed, we have $f'(x) = 2x - x^2$. The fixed-step-size gradient algorithm with step size $\alpha = 1/2$ is therefore given by

$$x^{(k+1)} = x^{(k)} - \alpha f'(x^{(k)}) = \frac{1}{2}(x^{(k)})^2.$$

With $x^{(0)} = 1$, we can derive the expression $x^{(k)} = (1/2)^{2^k-1}$. Hence, the algorithm converges to 0, a strict local minimizer of f .

Next, we find the order of convergence. Note that

$$\frac{|x^{(k+1)}|}{|x^{(k)}|^2} = \frac{1}{2}.$$

Therefore, the order of convergence is 2. ■

Finally, we show that the steepest descent algorithm has an order of convergence of 1 in the *worst case*; that is, there are cases for which the order of convergence of the steepest descent algorithm is equal to 1. To proceed, we will need the following simple lemma.

Lemma 8.3 *In the steepest descent algorithm, if $\mathbf{g}^{(k)} \neq \mathbf{0}$ for all k , then $\gamma_k = 1$ if and only if $\mathbf{g}^{(k)}$ is an eigenvector of \mathbf{Q} .* □

Proof. Suppose that $\mathbf{g}^{(k)} \neq \mathbf{0}$ for all k . Recall that for the steepest descent algorithm,

$$\gamma_k = \frac{(\mathbf{g}^{(k)\top} \mathbf{g}^{(k)})^2}{(\mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)})(\mathbf{g}^{(k)\top} \mathbf{Q}^{-1} \mathbf{g}^{(k)})}.$$

Sufficiency is easy to show by verification. To show necessity, suppose that $\gamma_k = 1$. Then, $V(\mathbf{x}^{(k+1)}) = 0$, which implies that $\mathbf{x}^{(k+1)} = \mathbf{x}^*$. Therefore,

$$\mathbf{x}^* = \mathbf{x}^{(k)} - \alpha_k \mathbf{g}^{(k)}.$$

Premultiplying by \mathbf{Q} and subtracting \mathbf{b} from both sides yields

$$\mathbf{0} = \mathbf{g}^{(k)} - \alpha_k \mathbf{Q} \mathbf{g}^{(k)},$$

which can be rewritten as

$$\mathbf{Q} \mathbf{g}^{(k)} = \frac{1}{\alpha_k} \mathbf{g}^{(k)}.$$

Hence, $\mathbf{g}^{(k)}$ is an eigenvector of \mathbf{Q} . ■

By the lemma, if $\mathbf{g}^{(k)}$ is not an eigenvector of \mathbf{Q} , then $\gamma_k < 1$ (recall that γ_k cannot exceed 1). We use this fact in the proof of the following result on the worst-case order of convergence of the steepest descent algorithm.

Theorem 8.6 *Let $\{\mathbf{x}^{(k)}\}$ be a convergent sequence of iterates of the steepest descent algorithm applied to a function f . Then, the order of convergence of $\{\mathbf{x}^{(k)}\}$ is 1 in the worst case; that is, there exist a function f and an initial condition $\mathbf{x}^{(0)}$ such that the order of convergence of $\{\mathbf{x}^{(k)}\}$ is equal to 1. \square*

Proof. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a quadratic function with Hessian \mathbf{Q} . Assume that the maximum and minimum eigenvalues of \mathbf{Q} satisfy $\lambda_{\max}(\mathbf{Q}) > \lambda_{\min}(\mathbf{Q})$. To show that the order of convergence of $\{\mathbf{x}^{(k)}\}$ is 1, it suffices to show that there exists $\mathbf{x}^{(0)}$ such that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \geq c \|\mathbf{x}^{(k)} - \mathbf{x}^*\|$$

for some $c > 0$ (see Exercise 8.2). Indeed, by Rayleigh's inequality,

$$\begin{aligned} V(\mathbf{x}^{(k+1)}) &= \frac{1}{2} (\mathbf{x}^{(k+1)} - \mathbf{x}^*)^\top \mathbf{Q} (\mathbf{x}^{(k+1)} - \mathbf{x}^*) \\ &\leq \frac{\lambda_{\max}(\mathbf{Q})}{2} \|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|^2. \end{aligned}$$

Similarly,

$$V(\mathbf{x}^{(k)}) \geq \frac{\lambda_{\min}(\mathbf{Q})}{2} \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2.$$

Combining the inequalities above with Lemma 8.1, we obtain

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \geq \sqrt{(1 - \gamma_k) \frac{\lambda_{\min}(\mathbf{Q})}{\lambda_{\max}(\mathbf{Q})}} \|\mathbf{x}^{(k)} - \mathbf{x}^*\|.$$

Therefore, it suffices to choose $\mathbf{x}^{(0)}$ such that $\gamma_k \leq d$ for some $d < 1$.

Recall that for the steepest descent algorithm, assuming that $\mathbf{g}^{(k)} \neq \mathbf{0}$ for all k , γ_k depends on $\mathbf{g}^{(k)}$ according to

$$\gamma_k = \frac{(\mathbf{g}^{(k)\top} \mathbf{g}^{(k)})^2}{(\mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)})(\mathbf{g}^{(k)\top} \mathbf{Q}^{-1} \mathbf{g}^{(k)})}.$$

First consider the case where $n = 2$. Suppose that $\mathbf{x}^{(0)} \neq \mathbf{x}^*$ is chosen such that $\mathbf{x}^{(0)} - \mathbf{x}^*$ is not an eigenvector of \mathbf{Q} . Then, $\mathbf{g}^{(0)} = \mathbf{Q}(\mathbf{x}^{(0)} - \mathbf{x}^*) \neq \mathbf{0}$ is also not an eigenvector of \mathbf{Q} . By Proposition 8.1, $\mathbf{g}^{(k)} = (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})/\alpha_k$ is not an eigenvector of \mathbf{Q} for any k [because any two eigenvectors corresponding to $\lambda_{\max}(\mathbf{Q})$ and $\lambda_{\min}(\mathbf{Q})$ are mutually orthogonal]. Also, $\mathbf{g}^{(k)}$ lies in one of two mutually orthogonal directions. Therefore, by Lemma 8.3, for each k , the value of γ_k is one of two numbers, both of which are strictly less than 1. This proves the $n = 2$ case.

For the general n case, let \mathbf{v}_1 and \mathbf{v}_2 be mutually orthogonal eigenvectors corresponding to $\lambda_{\max}(\mathbf{Q})$ and $\lambda_{\min}(\mathbf{Q})$. Choose $\mathbf{x}^{(0)}$ such that $\mathbf{x}^{(0)} - \mathbf{x}^* \neq \mathbf{0}$ lies in the span of \mathbf{v}_1 and \mathbf{v}_2 but is not equal to either. Note that $\mathbf{g}^{(0)} = \mathbf{Q}(\mathbf{x}^{(0)} - \mathbf{x}^*)$ also lies in the span of \mathbf{v}_1 and \mathbf{v}_2 , but is not equal to either. By manipulating $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{g}^{(k)}$ as before, we can write $\mathbf{g}^{(k+1)} = (\mathbf{I} - \alpha_k \mathbf{Q})\mathbf{g}^{(k)}$. Any eigenvector of \mathbf{Q} is also an eigenvector of $\mathbf{I} - \alpha_k \mathbf{Q}$. Therefore, $\mathbf{g}^{(k)}$ lies in the span of \mathbf{v}_1 and \mathbf{v}_2 for all k ; that is, the sequence $\{\mathbf{g}^{(k)}\}$ is confined within the two-dimensional subspace spanned by \mathbf{v}_1 and \mathbf{v}_2 . We can now proceed as in the $n = 2$ case. ■

In the next chapter we discuss Newton's method, which has order of convergence at least 2 if the initial guess is near the solution.

EXERCISES

8.1 Perform two iterations leading to the minimization of

$$f(x_1, x_2) = x_1 + \frac{1}{2}x_2 + \frac{1}{2}x_1^2 + x_2^2 + 3$$

using the steepest descent method with the starting point $\mathbf{x}^{(0)} = \mathbf{0}$. Also determine an optimal solution analytically.

8.2 Let $\{\mathbf{x}^{(k)}\}$ be a sequence that converges to \mathbf{x}^* . Show that if there exists $c > 0$ such that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \geq c\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^p$$

for sufficiently large k , then the order of convergence (if it exists) is at most p .

8.3 Let $\{\mathbf{x}^{(k)}\}$ be a sequence that converges to \mathbf{x}^* . Show that there does not exist $p < 1$ such that

$$\lim_{k \rightarrow \infty} \frac{\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|}{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^p} > 0.$$

8.4 Consider the sequence $\{x^{(k)}\}$ given by $x^{(k)} = 2^{-2^{k^2}}$.

- a. Write down the value of the limit of $\{x^{(k)}\}$.
- b. Find the order of convergence of $\{x^{(k)}\}$.

8.5 Consider the two sequences $\{x^{(k)}\}$ and $\{y^{(k)}\}$ defined iteratively as follows:

$$\begin{aligned}x^{(k+1)} &= ax^{(k)}, \\y^{(k+1)} &= (y^{(k)})^b,\end{aligned}$$

where $a \in \mathbb{R}$, $b \in \mathbb{R}$, $0 < a < 1$, $b > 1$, $x^{(0)} \neq 0$, $y^{(0)} \neq 0$, and $|y^{(0)}| < 1$.

- a. Derive a formula for $x^{(k)}$ in terms of $x^{(0)}$ and a . Use this to deduce that $x^{(k)} \rightarrow 0$.
- b. Derive a formula for $y^{(k)}$ in terms of $y^{(0)}$ and b . Use this to deduce that $y^{(k)} \rightarrow 0$.
- c. Find the order of convergence of $\{x^{(k)}\}$ and the order of convergence of $\{y^{(k)}\}$.
- d. Calculate the smallest number of iterations k such that $|x^{(k)}| \leq c|x^{(0)}|$, where $0 < c < 1$.
Hint: The answer is in terms of a and c . You may use the notation $[z]$ to represent the smallest integer not smaller than z .
- e. Calculate the smallest number of iterations k such that $|y^{(k)}| \leq c|y^{(0)}|$, where $0 < c < 1$.
- f. Compare the answer of part e with that of part d, focusing on the case where c is very small.

8.6 Suppose that we use the golden section algorithm to find the minimizer of a function. Let u_k be the uncertainty range at the k th iteration. Find the order of convergence of $\{u_k\}$.

8.7 Suppose that we wish to minimize a function $f : \mathbb{R} \rightarrow \mathbb{R}$ that has a derivative f' . A simple line search method, called *derivative descent search* (DDS), is described as follows: given that we are at a point $x^{(k)}$, we move in the direction of the negative derivative with step size α ; that is, $x^{(k+1)} = x^{(k)} - \alpha f'(x^{(k)})$, where $\alpha > 0$ is a constant.

In the following parts, assume that f is quadratic: $f(x) = \frac{1}{2}ax^2 - bx + c$ (where a , b , and c are constants, and $a > 0$).

- a. Write down the value of x^* (in terms of a , b , and c) that minimizes f .

- b. Write down the recursive equation for the DDS algorithm explicitly for this quadratic f .
- c. Assuming that the DDS algorithm converges, show that it converges to the optimal value x^* (found in part a).
- d. Find the order of convergence of the algorithm, assuming that it does converge.
- e. Find the range of values of α for which the algorithm converges (for this particular f) for all starting points $x^{(0)}$.

8.8 Consider the function

$$f(\mathbf{x}) = 3(x_1^2 + x_2^2) + 4x_1x_2 + 5x_1 + 6x_2 + 7,$$

where $\mathbf{x} = [x_1, x_2]^\top \in \mathbb{R}^2$. Suppose that we use a fixed-step-size gradient algorithm to find the minimizer of f :

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)}).$$

Find the largest range of values of α for which the algorithm is globally convergent.

8.9 This exercise explores a zero-finding algorithm.

Suppose that we wish to solve the equation $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, where

$$\mathbf{h}(\mathbf{x}) = \begin{bmatrix} 4 + 3x_1 + 2x_2 \\ 1 + 2x_1 + 3x_2 \end{bmatrix}.$$

Consider using an algorithm of the form $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \mathbf{h}(\mathbf{x}^{(k)})$, where α is scalar constant that does not depend on k .

- a. Find the solution of $\mathbf{h}(\mathbf{x}) = \mathbf{0}$.
- b. Find the largest range of values of α such that the algorithm is globally convergent to the solution of $\mathbf{h}(\mathbf{x}) = \mathbf{0}$.
- c. Assuming that α is outside the range of values in part b, give an example of an initial condition $\mathbf{x}^{(0)}$ of the form $[x_1, 0]^\top$ such that the algorithm is guaranteed not to satisfy the descent property.

8.10 Consider the function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ given by

$$f(\mathbf{x}) = \frac{3}{2}(x_1^2 + x_2^2) + (1+a)x_1x_2 - (x_1 + x_2) + b,$$

where a and b are some unknown real-valued parameters.

- Write the function f in the usual multivariable quadratic form.
- Find the largest set of values of a and b such that the unique global minimizer of f exists, and write down the minimizer (in terms of the parameters a and b).
- Consider the following algorithm:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \frac{2}{5} \nabla f(\mathbf{x}^{(k)}).$$

Find the largest set of values of a and b for which this algorithm converges to the global minimizer of f for any initial point $\mathbf{x}^{(0)}$.

8.11 Consider the function $f : \mathbb{R} \rightarrow \mathbb{R}$ given by $f(x) = \frac{1}{2}(x - c)^2$, $c \in \mathbb{R}$. We are interested in computing the minimizer of f using the iterative algorithm

$$x^{(k+1)} = x^{(k)} - \alpha_k f'(x^{(k)}),$$

where f' is the derivative of f and α_k is a step size satisfying $0 < \alpha_k < 1$.

- Derive a formula relating $f(x^{(k+1)})$ with $f(x^{(k)})$, involving α_k .
- Show that the algorithm is globally convergent if and only if

$$\sum_{k=0}^{\infty} \alpha_k = \infty.$$

Hint: Use part a and the fact that for any sequence $\{\alpha_k\} \subset (0, 1)$, we have

$$\prod_{k=0}^{\infty} (1 - \alpha_k) = 0 \Leftrightarrow \sum_{k=0}^{\infty} \alpha_k = \infty.$$

8.12 Consider the function $f : \mathbb{R} \rightarrow \mathbb{R}$ given by $f(x) = x^3 - x$. Suppose that we use a fixed-step-size algorithm $x^{(k+1)} = x^{(k)} - \alpha f'(x^{(k)})$ to find a local minimizer of f . Find the largest range of values of α such that the algorithm is locally convergent (i.e., for all x_0 sufficiently close to a local minimizer x^* , we have $x^{(k)} \rightarrow x^*$).

8.13 Consider the function f given by $f(x) = (x - 1)^2$, $x \in \mathbb{R}$. We are interested in computing the minimizer of f using the iterative algorithm $x^{(k+1)} = x^{(k)} - \alpha 2^{-k} f'(x^{(k)})$, where f' is the derivative of f and $0 < \alpha < 1$. Does the algorithm have the descent property? Is the algorithm globally convergent?

8.14 Let $f : \mathbb{R} \rightarrow \mathbb{R}$, $f \in \mathcal{C}^3$, with first derivative f' , second derivative f'' , and unique minimizer x^* . Consider a fixed-step-size gradient algorithm

$$x^{(k+1)} = x^{(k)} - \alpha f'(x^{(k)}).$$

Suppose that $f''(x^*) \neq 0$ and $\alpha = 1/f''(x^*)$. Assuming that the algorithm converges to x^* , show that the order of convergence is at least 2.

8.15 Consider the problem of minimizing $f(x) = \|\mathbf{a}x - \mathbf{b}\|^2$, where \mathbf{a} and \mathbf{b} are vectors in \mathbb{R}^n , and $\mathbf{a} \neq \mathbf{0}$.

- a. Derive an expression (in terms of \mathbf{a} and \mathbf{b}) for the solution to this problem.
- b. To solve the problem, suppose that we use an iterative algorithm of the form

$$x^{(k+1)} = x^{(k)} - \alpha f'(x^{(k)}),$$

where f' is the derivative of f . Find the largest range of values of α (in terms of \mathbf{a} and \mathbf{b}) for which the algorithm converges to the solution for all starting points $x^{(0)}$.

8.16 Consider the optimization problem

$$\text{minimize } \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2,$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \geq n$, and $\mathbf{b} \in \mathbb{R}^m$.

- a. Show that the objective function for this problem is a quadratic function, and write down the gradient and Hessian of this quadratic.
- b. Write down the fixed-step-size gradient algorithm for solving this optimization problem.
- c. Suppose that

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}.$$

Find the largest range of values for α such that the algorithm in part b converges to the solution of the problem.

8.17 Consider a function $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ given by $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{b} \in \mathbb{R}^n$. Suppose that \mathbf{A} is invertible and \mathbf{x}^* is the zero of \mathbf{f} [i.e., $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$]. We wish to compute \mathbf{x}^* using the iterative algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \mathbf{f}(\mathbf{x}^{(k)}),$$

where $\alpha \in \mathbb{R}$, $\alpha > 0$. We say that the algorithm is *globally monotone* if for any $\mathbf{x}^{(0)}$, $\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq \|\mathbf{x}^{(k)} - \mathbf{x}^*\|$ for all k .

- a. Assume that all the eigenvalues of \mathbf{A} are real. Show that a necessary condition for the algorithm above to be *globally monotone* is that all the eigenvalues of \mathbf{A} are nonnegative.

Hint: Use contraposition.

- b. Suppose that

$$\mathbf{A} = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}.$$

Find the largest range of values of α for which the algorithm is *globally convergent* (i.e., $\mathbf{x}^{(k)} \rightarrow \mathbf{x}^*$ for all $\mathbf{x}^{(0)}$).

- 8.18** Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be given by $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b}$, where $\mathbf{b} \in \mathbb{R}^n$ and \mathbf{Q} is a real symmetric positive definite $n \times n$ matrix. Suppose that we apply the steepest descent method to this function, with $\mathbf{x}^{(0)} \neq \mathbf{Q}^{-1}\mathbf{b}$. Show that the method converges in one step, that is, $\mathbf{x}^{(1)} = \mathbf{Q}^{-1}\mathbf{b}$, if and only if $\mathbf{x}^{(0)}$ is chosen such that $\mathbf{g}^{(0)} = \mathbf{Q}\mathbf{x}^{(0)} - \mathbf{b}$ is an eigenvector of \mathbf{Q} .

- 8.19** Suppose that we apply the steepest descent algorithm $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{g}^{(k)}$ to a quadratic function f with Hessian $\mathbf{Q} > 0$. Let λ_{\max} and λ_{\min} be the largest and smallest eigenvalue of \mathbf{Q} , respectively. Which of the following two inequalities are possibly true? (When we say here that an inequality is “possibly” true, we mean that there exists a choice of f and $\mathbf{x}^{(0)}$ such that the inequality holds.)

- a. $\alpha_0 \geq 2/\lambda_{\max}$.
- b. $\alpha_0 > 1/\lambda_{\min}$.

- 8.20** Suppose that we apply a fixed-step-size gradient algorithm to minimize

$$f(\mathbf{x}) = \mathbf{x}^\top \begin{bmatrix} 3/2 & 2 \\ 0 & 3/2 \end{bmatrix} \mathbf{x} + \mathbf{x}^\top \begin{bmatrix} 3 \\ -1 \end{bmatrix} - 22.$$

- a. Find the range of values of the step size for which the algorithm converges to the minimizer.
- b. Suppose that we use a step size of 1000 (which is too large). Find an initial condition that will cause the algorithm to diverge (not converge).

- 8.21** Consider a fixed-step-size gradient algorithm applied to each of the functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ in parts a and b below. In each case, find the largest range of values of the step size α for which the algorithm is globally convergent.

- a. $f(\mathbf{x}) = 1 + 2x_1 + 3(x_1^2 + x_2^2) + 4x_1x_2$.

b. $f(\mathbf{x}) = \mathbf{x}^\top \begin{bmatrix} 3 & 3 \\ 1 & 3 \end{bmatrix} \mathbf{x} + [16, 23]\mathbf{x} + \pi^2.$

8.22 Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be given by $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b}$, where $\mathbf{b} \in \mathbb{R}^n$ and \mathbf{Q} is a real symmetric positive definite $n \times n$ matrix. Consider the algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \beta \alpha_k \mathbf{g}^{(k)},$$

where $\mathbf{g}^{(k)} = \mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b}$, $\alpha_k = \mathbf{g}^{(k)\top} \mathbf{g}^{(k)} / \mathbf{g}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)}$, and $\beta \in \mathbb{R}$ is a given constant. (Note that the above reduces to the steepest descent algorithm if $\beta = 1$.) Show that $\{\mathbf{x}^{(k)}\}$ converges to $\mathbf{x}^* = \mathbf{Q}^{-1}\mathbf{b}$ for any initial condition $\mathbf{x}^{(0)}$ if and only if $0 < \beta < 2$.

8.23 Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be given by $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b}$, where $\mathbf{b} \in \mathbb{R}^n$ and \mathbf{Q} is a real symmetric positive definite $n \times n$ matrix. Consider a gradient algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{g}^{(k)},$$

where $\mathbf{g}^{(k)} = \mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b}$ is the gradient of f at $\mathbf{x}^{(k)}$ and α_k is some step size. Show that the algorithm has the descent property [i.e., $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$ whenever $\mathbf{g}^{(k)} \neq \mathbf{0}$] if and only if $\gamma_k > 0$ for all k .

8.24 Given $f : \mathbb{R}^n \rightarrow \mathbb{R}$, consider the general iterative algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},$$

where $\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \dots$ are given vectors in \mathbb{R}^n and α_k is chosen to minimize $f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$; that is,

$$\alpha_k = \arg \min f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}).$$

Show that for each k , the vector $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ is orthogonal to $\nabla f(\mathbf{x}^{(k+1)})$ (assuming that the gradient exists).

8.25 Write a simple MATLAB program for implementing the steepest descent algorithm using the secant method for the line search (e.g., the MATLAB function of Exercise 7.11). For the stopping criterion, use the condition $\|\mathbf{g}^{(k)}\| \leq \varepsilon$, where $\varepsilon = 10^{-6}$. Test your program by comparing the output with the numbers in Example 8.1. Also test your program using an initial condition of $[-4, 5, 1]^\top$, and determine the number of iterations required to satisfy the stopping criterion. Evaluate the objective function at the final point to see how close it is to 0.

8.26 Apply the MATLAB program from Exercise 8.25 to Rosenbrock's function:

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Use an initial condition of $\mathbf{x}^{(0)} = [-2, 2]^\top$. Terminate the algorithm when the norm of the gradient of f is less than 10^{-4} .

CHAPTER 9

NEWTON'S METHOD

9.1 Introduction

Recall that the method of steepest descent uses only first derivatives (gradients) in selecting a suitable search direction. This strategy is not always the most effective. If higher derivatives are used, the resulting iterative algorithm may perform better than the steepest descent method. *Newton's method* (sometimes called the *Newton-Raphson method*) uses first and second derivatives and indeed does perform better than the steepest descent method if the initial point is close to the minimizer. The idea behind this method is as follows. Given a starting point, we construct a quadratic approximation to the objective function that matches the first and second derivative values at that point. We then minimize the approximate (quadratic) function instead of the original objective function. We use the minimizer of the approximate function as the starting point in the next step and repeat the procedure iteratively. If the objective function is quadratic, then the approximation is exact, and the method yields the true minimizer in one step. If, on the other hand, the objective function is not quadratic, then the approximation will provide

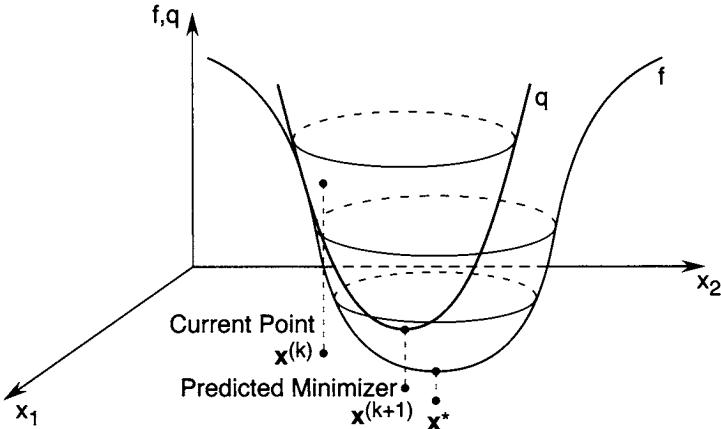


Figure 9.1 Quadratic approximation to the objective function using first and second derivatives.

only an estimate of the position of the true minimizer. Figure 9.1 illustrates this idea.

We can obtain a quadratic approximation to the twice continuously differentiable objection function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ using the Taylor series expansion of f about the current point $\mathbf{x}^{(k)}$, neglecting terms of order three and higher. We obtain

$$f(\mathbf{x}) \approx f(\mathbf{x}^{(k)}) + (\mathbf{x} - \mathbf{x}^{(k)})^\top \mathbf{g}^{(k)} + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(k)})^\top \mathbf{F}(\mathbf{x}^{(k)}) (\mathbf{x} - \mathbf{x}^{(k)}) \triangleq q(\mathbf{x}),$$

where, for simplicity, we use the notation $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$. Applying the FONC to q yields

$$\mathbf{0} = \nabla q(\mathbf{x}) = \mathbf{g}^{(k)} + \mathbf{F}(\mathbf{x}^{(k)}) (\mathbf{x} - \mathbf{x}^{(k)}).$$

If $\mathbf{F}(\mathbf{x}^{(k)}) > 0$, then q achieves a minimum at

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)}.$$

This recursive formula represents Newton's method.

Example 9.1 Use Newton's method to minimize the Powell function:

$$f(x_1, x_2, x_3, x_4) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4.$$

Use as the starting point $\mathbf{x}^{(0)} = [3, -1, 0, 1]^\top$. Perform three iterations.

Note that $f(\mathbf{x}^{(0)}) = 215$. We have

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2(x_1 + 10x_2) + 40(x_1 - x_4)^3 \\ 20(x_1 + 10x_2) + 4(x_2 - 2x_3)^3 \\ 10(x_3 - x_4) - 8(x_2 - 2x_3)^3 \\ -10(x_3 - x_4) - 40(x_1 - x_4)^3 \end{bmatrix},$$

and $\mathbf{F}(\mathbf{x})$ is given by

$$\begin{bmatrix} 2 + 120(x_1 - x_4)^2 & 20 & 0 & -120(x_1 - x_4)^2 \\ 20 & 200 + 12(x_2 - 2x_3)^2 & -24(x_2 - 2x_3)^2 & 0 \\ 0 & -24(x_2 - 2x_3)^2 & 10 + 48(x_2 - 2x_3)^2 & -10 \\ -120(x_1 - x_4)^2 & 0 & -10 & 10 + 120(x_1 - x_4)^2 \end{bmatrix}.$$

Iteration 1

$$\begin{aligned} \mathbf{g}^{(0)} &= [306, -144, -2, -310]^\top, \\ \mathbf{F}(\mathbf{x}^{(0)}) &= \begin{bmatrix} 482 & 20 & 0 & -480 \\ 20 & 212 & -24 & 0 \\ 0 & -24 & 58 & -10 \\ -480 & 0 & -10 & 490 \end{bmatrix}, \\ \mathbf{F}(\mathbf{x}^{(0)})^{-1} &= \begin{bmatrix} 0.1126 & -0.0089 & 0.0154 & 0.1106 \\ -0.0089 & 0.0057 & 0.0008 & -0.0087 \\ 0.0154 & 0.0008 & 0.0203 & 0.0155 \\ 0.1106 & -0.0087 & 0.0155 & 0.1107 \end{bmatrix}, \\ \mathbf{F}(\mathbf{x}^{(0)})^{-1}\mathbf{g}^{(0)} &= [1.4127, -0.8413, -0.2540, 0.7460]^\top. \end{aligned}$$

Hence,

$$\begin{aligned} \mathbf{x}^{(1)} &= \mathbf{x}^{(0)} - \mathbf{F}(\mathbf{x}^{(0)})^{-1}\mathbf{g}^{(0)} = [1.5873, -0.1587, 0.2540, 0.2540]^\top, \\ f(\mathbf{x}^{(1)}) &= 31.8. \end{aligned}$$

Iteration 2

$$\begin{aligned} \mathbf{g}^{(1)} &= [94.81, -1.179, 2.371, -94.81]^\top, \\ \mathbf{F}(\mathbf{x}^{(1)}) &= \begin{bmatrix} 215.3 & 20 & 0 & -213.3 \\ 20 & 205.3 & -10.67 & 0 \\ 0 & -10.67 & 31.34 & -10 \\ -213.3 & 0 & -10 & 223.3 \end{bmatrix}, \\ \mathbf{F}(\mathbf{x}^{(1)})^{-1}\mathbf{g}^{(1)} &= [0.5291, -0.0529, 0.0846, 0.0846]^\top. \end{aligned}$$

Hence,

$$\begin{aligned} \mathbf{x}^{(2)} &= \mathbf{x}^{(1)} - \mathbf{F}(\mathbf{x}^{(1)})^{-1}\mathbf{g}^{(1)} = [1.0582, -0.1058, 0.1694, 0.1694]^\top, \\ f(\mathbf{x}^{(2)}) &= 6.28. \end{aligned}$$

Iteration 3

$$\begin{aligned}\mathbf{g}^{(2)} &= [28.09, -0.3475, 0.7031, -28.08]^\top, \\ \mathbf{F}(\mathbf{x}^{(2)}) &= \begin{bmatrix} 96.80 & 20 & 0 & -94.80 \\ 20 & 202.4 & -4.744 & 0 \\ 0 & -4.744 & 19.49 & -10 \\ -94.80 & 0 & -10 & 104.80 \end{bmatrix}, \\ \mathbf{x}^{(3)} &= [0.7037, -0.0704, 0.1121, 0.1111]^\top, \\ f(\mathbf{x}^{(3)}) &= 1.24.\end{aligned}$$

■

Observe that the k th iteration of Newton's method can be written in two steps as

1. Solve $\mathbf{F}(\mathbf{x}^{(k)})\mathbf{d}^{(k)} = -\mathbf{g}^{(k)}$ for $\mathbf{d}^{(k)}$.
2. Set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$.

Step 1 requires the solution of an $n \times n$ system of linear equations. Thus, an efficient method for solving systems of linear equations is essential when using Newton's method.

As in the one-variable case, Newton's method can also be viewed as a technique for iteratively solving the equation

$$\mathbf{g}(\mathbf{x}) = \mathbf{0},$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^n$. In this case $\mathbf{F}(\mathbf{x})$ is the Jacobian matrix of \mathbf{g} at \mathbf{x} ; that is, $\mathbf{F}(\mathbf{x})$ is the $n \times n$ matrix whose (i, j) entry is $(\partial g_i / \partial x_j)(\mathbf{x})$, $i, j = 1, 2, \dots, n$.

9.2 Analysis of Newton's Method

As in the one-variable case there is no guarantee that Newton's algorithm heads in the direction of decreasing values of the objective function if $\mathbf{F}(\mathbf{x}^{(k)})$ is not positive definite (recall Figure 7.7 illustrating Newton's method for functions of one variable when $f'' < 0$). Moreover, even if $\mathbf{F}(\mathbf{x}^{(k)}) > 0$, Newton's method may not be a descent method; that is, it is possible that $f(\mathbf{x}^{(k+1)}) \geq f(\mathbf{x}^{(k)})$. For example, this may occur if our starting point $\mathbf{x}^{(0)}$ is far away from the solution. See the end of this section for a possible remedy to this problem. Despite these drawbacks, Newton's method has superior convergence properties when the starting point is near the solution, as we shall see in the remainder of this section.

The convergence analysis of Newton's method when f is a quadratic function is straightforward. In fact, Newton's method reaches the point \mathbf{x}^* such

that $\nabla f(\mathbf{x}^*) = \mathbf{0}$ in just one step starting from any initial point $\mathbf{x}^{(0)}$. To see this, suppose that $\mathbf{Q} = \mathbf{Q}^\top$ is invertible and

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b}.$$

Then,

$$\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x}) = \mathbf{Q}\mathbf{x} - \mathbf{b}$$

and

$$\mathbf{F}(\mathbf{x}) = \mathbf{Q}.$$

Hence, given any initial point $\mathbf{x}^{(0)}$, by Newton's algorithm

$$\begin{aligned}\mathbf{x}^{(1)} &= \mathbf{x}^{(0)} - \mathbf{F}(\mathbf{x}^{(0)})^{-1}\mathbf{g}^{(0)} \\ &= \mathbf{x}^{(0)} - \mathbf{Q}^{-1}[\mathbf{Q}\mathbf{x}^{(0)} - \mathbf{b}] \\ &= \mathbf{Q}^{-1}\mathbf{b} \\ &= \mathbf{x}^*.\end{aligned}$$

Therefore, for the quadratic case the order of convergence of Newton's algorithm is ∞ for any initial point $\mathbf{x}^{(0)}$ (compare this with Exercise 8.18, which deals with the steepest descent algorithm).

To analyze the convergence of Newton's method in the general case, we use results from Section 5.1. Let $\{\mathbf{x}^{(k)}\}$ be the Newton's method sequence for minimizing a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We show that $\{\mathbf{x}^{(k)}\}$ converges to the minimizer \mathbf{x}^* with order of convergence at least 2.

Theorem 9.1 Suppose that $f \in \mathcal{C}^3$ and $\mathbf{x}^* \in \mathbb{R}^n$ is a point such that $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\mathbf{F}(\mathbf{x}^*)$ is invertible. Then, for all $\mathbf{x}^{(0)}$ sufficiently close to \mathbf{x}^* , Newton's method is well-defined for all k and converges to \mathbf{x}^* with an order of convergence at least 2. \square

Proof. The Taylor series expansion of ∇f about $\mathbf{x}^{(0)}$ yields

$$\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}^{(0)}) - \mathbf{F}(\mathbf{x}^{(0)})(\mathbf{x} - \mathbf{x}^{(0)}) = O(\|\mathbf{x} - \mathbf{x}^{(0)}\|^2).$$

Because by assumption $f \in \mathcal{C}^3$ and $\mathbf{F}(\mathbf{x}^*)$ is invertible, there exist constants $\varepsilon > 0$, $c_1 > 0$, and $c_2 > 0$ such that if $\mathbf{x}^{(0)}, \mathbf{x} \in \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\| \leq \varepsilon\}$, we have

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}^{(0)}) - \mathbf{F}(\mathbf{x}^{(0)})(\mathbf{x} - \mathbf{x}^{(0)})\| \leq c_1 \|\mathbf{x} - \mathbf{x}^{(0)}\|^2$$

and by Lemma 5.3, $\mathbf{F}(\mathbf{x})^{-1}$ exists and satisfies

$$\|\mathbf{F}(\mathbf{x})^{-1}\| \leq c_2.$$

The first inequality above holds because the remainder term in the Taylor series expansion contains third derivatives of f that are continuous and hence bounded on $\{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\| \leq \varepsilon\}$.

Suppose that $\mathbf{x}^{(0)} \in \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\| \leq \varepsilon\}$. Then, substituting $\mathbf{x} = \mathbf{x}^*$ in the inequality above and using the assumption that $\nabla f(\mathbf{x}^*) = \mathbf{0}$, we get

$$\|\mathbf{F}(\mathbf{x}^{(0)})(\mathbf{x}^{(0)} - \mathbf{x}^*) - \nabla f(\mathbf{x}^{(0)})\| \leq c_1 \|\mathbf{x}^{(0)} - \mathbf{x}^*\|^2.$$

Now, subtracting \mathbf{x}^* from both sides of Newton's algorithm and taking norms yields

$$\begin{aligned}\|\mathbf{x}^{(1)} - \mathbf{x}^*\| &= \|\mathbf{x}^{(0)} - \mathbf{x}^* - \mathbf{F}(\mathbf{x}^{(0)})^{-1} \nabla f(\mathbf{x}^{(0)})\| \\ &= \|\mathbf{F}(\mathbf{x}^{(0)})^{-1} (\mathbf{F}(\mathbf{x}^{(0)})(\mathbf{x}^{(0)} - \mathbf{x}^*) - \nabla f(\mathbf{x}^{(0)}))\| \\ &\leq \|\mathbf{F}(\mathbf{x}^{(0)})^{-1}\| \|\mathbf{F}(\mathbf{x}^{(0)})(\mathbf{x}^{(0)} - \mathbf{x}^*) - \nabla f(\mathbf{x}^{(0)})\|.\end{aligned}$$

Applying the inequalities above involving the constants c_1 and c_2 gives

$$\|\mathbf{x}^{(1)} - \mathbf{x}^*\| \leq c_1 c_2 \|\mathbf{x}^{(0)} - \mathbf{x}^*\|^2.$$

Suppose that $\mathbf{x}^{(0)}$ is such that

$$\|\mathbf{x}^{(0)} - \mathbf{x}^*\| \leq \frac{\alpha}{c_1 c_2},$$

where $\alpha \in (0, 1)$. Then,

$$\|\mathbf{x}^{(1)} - \mathbf{x}^*\| \leq \alpha \|\mathbf{x}^{(0)} - \mathbf{x}^*\|.$$

By induction, we obtain

$$\begin{aligned}\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| &\leq c_1 c_2 \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2, \\ \|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| &\leq \alpha \|\mathbf{x}^{(k)} - \mathbf{x}^*\|.\end{aligned}$$

Hence,

$$\lim_{k \rightarrow \infty} \|\mathbf{x}^{(k)} - \mathbf{x}^*\| = 0,$$

and therefore the sequence $\{\mathbf{x}^{(k)}\}$ converges to \mathbf{x}^* . The order of convergence is at least 2 because $\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq c_1 c_2 \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2$; that is, $\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| = O(\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2)$. ■

Warning: In the Theorem 9.1, we did not state that \mathbf{x}^* is a local minimizer. For example, if \mathbf{x}^* is a local *maximizer*, then provided that $f \in C^3$ and $\mathbf{F}(\mathbf{x}^*)$ is invertible, Newton's method would converge to \mathbf{x}^* if we start close enough to it.

As stated in Theorem 9.1, Newton's method has superior convergence properties if the starting point is near the solution. However, the method is not guaranteed to converge to the solution if we start far away from it (in fact, it may not even be well-defined because the Hessian may be singular). In particular, the method may not be a descent method; that is, it is possible that

$f(\mathbf{x}^{(k+1)}) \geq f(\mathbf{x}^{(k)})$. Fortunately, it is possible to modify the algorithm such that the descent property holds. To see this, we need the following result.

Theorem 9.2 *Let $\{\mathbf{x}^{(k)}\}$ be the sequence generated by Newton's method for minimizing a given objective function $f(\mathbf{x})$. If the Hessian $\mathbf{F}(\mathbf{x}^{(k)}) > 0$ and $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)}) \neq \mathbf{0}$, then the search direction*

$$\mathbf{d}^{(k)} = -\mathbf{F}(\mathbf{x}^{(k)})^{-1}\mathbf{g}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$$

from $\mathbf{x}^{(k)}$ to $\mathbf{x}^{(k+1)}$ is a descent direction for f in the sense that there exists an $\bar{\alpha} > 0$ such that for all $\alpha \in (0, \bar{\alpha})$,

$$f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)}).$$

□

Proof. Let

$$\phi(\alpha) = f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}).$$

Then, using the chain rule, we obtain

$$\phi'(\alpha) = \nabla f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)})^\top \mathbf{d}^{(k)}.$$

Hence,

$$\phi'(0) = \nabla f(\mathbf{x}^{(k)})^\top \mathbf{d}^{(k)} = -\mathbf{g}^{(k)\top} \mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)} < 0,$$

because $\mathbf{F}(\mathbf{x}^{(k)})^{-1} > 0$ and $\mathbf{g}^{(k)} \neq \mathbf{0}$. Thus, there exists an $\bar{\alpha} > 0$ so that for all $\alpha \in (0, \bar{\alpha})$, $\phi(\alpha) < \phi(0)$. This implies that for all $\alpha \in (0, \bar{\alpha})$,

$$f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)}),$$

which completes the proof. ■

Theorem 9.2 motivates the following modification of Newton's method:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)},$$

where

$$\alpha_k = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} - \alpha \mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)});$$

that is, at each iteration, we perform a line search in the direction $-\mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)}$. By Theorem 9.2 we conclude that the modified Newton's method has the descent property; that is,

$$f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$$

whenever $\mathbf{g}^{(k)} \neq \mathbf{0}$.

A drawback of Newton's method is that evaluation of $\mathbf{F}(\mathbf{x}^{(k)})$ for large n can be computationally expensive. Furthermore, we have to solve the set of n linear equations $\mathbf{F}(\mathbf{x}^{(k)})\mathbf{d}^{(k)} = -\mathbf{g}^{(k)}$. In Chapters 10 and 11 we discuss methods that alleviate this difficulty.

Another source of potential problems in Newton's method arises from the Hessian matrix not being positive definite. In the next section we describe a simple modification of Newton's method to overcome this problem.

9.3 Levenberg-Marquardt Modification

If the Hessian matrix $\mathbf{F}(\mathbf{x}^{(k)})$ is not positive definite, then the search direction $\mathbf{d}^{(k)} = -\mathbf{F}(\mathbf{x}^{(k)})^{-1}\mathbf{g}^{(k)}$ may not point in a descent direction. A simple technique to ensure that the search direction is a descent direction is to introduce the *Levenberg-Marquardt modification* of Newton's algorithm:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{F}(\mathbf{x}^{(k)}) + \mu_k \mathbf{I})^{-1} \mathbf{g}^{(k)},$$

where $\mu_k \geq 0$.

The idea underlying the Levenberg-Marquardt modification is as follows. Consider a symmetric matrix \mathbf{F} , which may not be positive definite. Let $\lambda_1, \dots, \lambda_n$ be the eigenvalues of \mathbf{F} with corresponding eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_n$. The eigenvalues $\lambda_1, \dots, \lambda_n$ are real, but may not all be positive. Next, consider the matrix $\mathbf{G} = \mathbf{F} + \mu \mathbf{I}$, where $\mu \geq 0$. Note that the eigenvalues of \mathbf{G} are $\lambda_1 + \mu, \dots, \lambda_n + \mu$. Indeed,

$$\begin{aligned} \mathbf{G}\mathbf{v}_i &= (\mathbf{F} + \mu \mathbf{I})\mathbf{v}_i \\ &= \mathbf{F}\mathbf{v}_i + \mu \mathbf{I}\mathbf{v}_i \\ &= \lambda_i \mathbf{v}_i + \mu \mathbf{v}_i \\ &= (\lambda_i + \mu) \mathbf{v}_i, \end{aligned}$$

which shows that for all $i = 1, \dots, n$, \mathbf{v}_i is also an eigenvector of \mathbf{G} with eigenvalue $\lambda_i + \mu$. Therefore, if μ is sufficiently large, then all the eigenvalues of \mathbf{G} are positive and \mathbf{G} is positive definite. Accordingly, if the parameter μ_k in the Levenberg-Marquardt modification of Newton's algorithm is sufficiently large, then the search direction $\mathbf{d}^{(k)} = -(\mathbf{F}(\mathbf{x}^{(k)}) + \mu_k \mathbf{I})^{-1} \mathbf{g}^{(k)}$ always points in a descent direction (in the sense of Theorem 9.2). In this case if we further introduce a step size α_k as described in Section 9.2,

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k (\mathbf{F}(\mathbf{x}^{(k)}) + \mu_k \mathbf{I})^{-1} \mathbf{g}^{(k)},$$

then we are guaranteed that the descent property holds.

The Levenberg-Marquardt modification of Newton's algorithm can be made to approach the behavior of the pure Newton's method by letting $\mu_k \rightarrow 0$. On the other hand, by letting $\mu_k \rightarrow \infty$, the algorithm approaches a pure gradient method with small step size. In practice, we may start with a small value of μ_k and increase it slowly until we find that the iteration is descent: $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$.

9.4 Newton's Method for Nonlinear Least Squares

We now examine a particular class of optimization problems and the use of Newton's method for solving them. Consider the following problem:

$$\text{minimize } \sum_{i=1}^m (r_i(\mathbf{x}))^2,$$

where $r_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, m$, are given functions. This particular problem is called a *nonlinear least-squares problem*. The special case where the r_i are linear is discussed in Section 12.1.

Example 9.2 Suppose that we are given m measurements of a process at m points in time, as depicted in Figure 9.2 (here, $m = 21$). Let t_1, \dots, t_m denote the measurement times and y_1, \dots, y_m the measurement values. Note that $t_1 = 0$ while $t_{21} = 10$. We wish to fit a sinusoid to the measurement data. The equation of the sinusoid is

$$y = A \sin(\omega t + \phi)$$

with appropriate choices of the parameters A , ω , and ϕ . To formulate the data-fitting problem, we construct the objective function

$$\sum_{i=1}^m (y_i - A \sin(\omega t_i + \phi))^2,$$

representing the sum of the squared errors between the measurement values and the function values at the corresponding points in time. Let $\mathbf{x} = [A, \omega, \phi]^\top$ represent the vector of decision variables. We therefore obtain a nonlinear least-squares problem with

$$r_i(\mathbf{x}) = y_i - A \sin(\omega t_i + \phi).$$

■

Defining $\mathbf{r} = [r_1, \dots, r_m]^\top$, we write the objective function as $f(\mathbf{x}) = \mathbf{r}(\mathbf{x})^\top \mathbf{r}(\mathbf{x})$. To apply Newton's method, we need to compute the gradient and the Hessian of f . The j th component of $\nabla f(\mathbf{x})$ is

$$(\nabla f(\mathbf{x}))_j = \frac{\partial f}{\partial x_j}(\mathbf{x}) = 2 \sum_{i=1}^m r_i(\mathbf{x}) \frac{\partial r_i}{\partial x_j}(\mathbf{x}).$$

Denote the Jacobian matrix of \mathbf{r} by

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial r_1}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial r_1}{\partial x_n}(\mathbf{x}) \\ \vdots & & \vdots \\ \frac{\partial r_m}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial r_m}{\partial x_n}(\mathbf{x}) \end{bmatrix}.$$

Then, the gradient of f can be represented as

$$\nabla f(\mathbf{x}) = 2\mathbf{J}(\mathbf{x})^\top \mathbf{r}(\mathbf{x}).$$

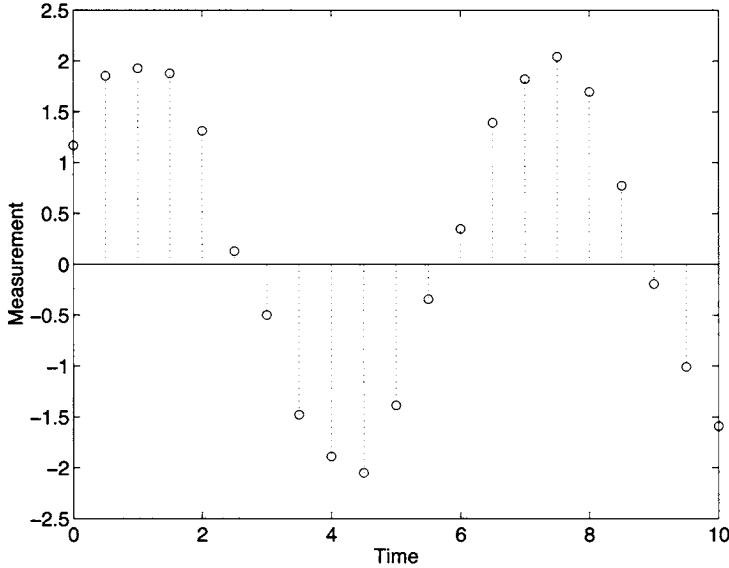


Figure 9.2 Measurement data for Example 9.2.

Next, we compute the Hessian matrix of f . The (k, j) th component of the Hessian is given by

$$\begin{aligned} \frac{\partial^2 f}{\partial x_k \partial x_j}(\mathbf{x}) &= \frac{\partial}{\partial x_k} \left(\frac{\partial f}{\partial x_j}(\mathbf{x}) \right) \\ &= \frac{\partial}{\partial x_k} \left(2 \sum_{i=1}^m r_i(\mathbf{x}) \frac{\partial r_i}{\partial x_j}(\mathbf{x}) \right) \\ &= 2 \sum_{i=1}^m \left(\frac{\partial r_i}{\partial x_k}(\mathbf{x}) \frac{\partial r_i}{\partial x_j}(\mathbf{x}) + r_i(\mathbf{x}) \frac{\partial^2 r_i}{\partial x_k \partial x_j}(\mathbf{x}) \right). \end{aligned}$$

Letting $\mathbf{S}(\mathbf{x})$ be the matrix whose (k, j) th component is

$$\sum_{i=1}^m r_i(\mathbf{x}) \frac{\partial^2 r_i}{\partial x_k \partial x_j}(\mathbf{x}),$$

we write the Hessian matrix as

$$\mathbf{F}(\mathbf{x}) = 2(\mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}) + \mathbf{S}(\mathbf{x})).$$

Therefore, Newton's method applied to the nonlinear least-squares problem is given by

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}) + \mathbf{S}(\mathbf{x}))^{-1} \mathbf{J}(\mathbf{x})^\top \mathbf{r}(\mathbf{x}).$$

In some applications, the matrix $\mathbf{S}(\mathbf{x})$ involving the second derivatives of the function \mathbf{r} can be ignored because its components are negligibly small. In this case Newton's algorithm reduces to what is commonly called the *Gauss-Newton method*:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}))^{-1} \mathbf{J}(\mathbf{x})^\top \mathbf{r}(\mathbf{x}).$$

Note that the Gauss-Newton method does not require calculation of the second derivatives of \mathbf{r} .

Example 9.3 Recall the data-fitting problem in Example 9.2, with

$$r_i(\mathbf{x}) = y_i - A \sin(\omega t_i + \phi), \quad i = 1, \dots, 21.$$

The Jacobian matrix $\mathbf{J}(\mathbf{x})$ in this problem is a 21×3 matrix with elements given by

$$\begin{aligned} (\mathbf{J}(\mathbf{x}))_{(i,1)} &= -\sin(\omega t_i + \phi), \\ (\mathbf{J}(\mathbf{x}))_{(i,2)} &= -t_i A \cos(\omega t_i + \phi), \\ (\mathbf{J}(\mathbf{x}))_{(i,3)} &= -A \cos(\omega t_i + \phi), \quad i = 1, \dots, 21. \end{aligned}$$

Using the expressions above, we apply the Gauss-Newton algorithm to find the sinusoid of best fit, given the data pairs $(t_1, y_1), \dots, (t_m, y_m)$. Figure 9.3 shows a plot of the sinusoid of best fit obtained from the Gauss-Newton algorithm. The parameters of this sinusoid are: $A = 2.01$, $\omega = 0.992$, and $\phi = 0.541$. ■

A potential problem with the Gauss-Newton method is that the matrix $\mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x})$ may not be positive definite. As described before, this problem can be overcome using a Levenberg-Marquardt modification:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}) + \mu_k \mathbf{I})^{-1} \mathbf{J}(\mathbf{x})^\top \mathbf{r}(\mathbf{x}).$$

This is referred to in the literature as the *Levenberg-Marquardt algorithm*, because the original Levenberg-Marquardt modification was developed specifically for the nonlinear least-squares problem. An alternative interpretation of the Levenberg-Marquardt algorithm is to view the term $\mu_k \mathbf{I}$ as an approximation to $\mathbf{S}(\mathbf{x})$ in Newton's algorithm.

EXERCISES

9.1 Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be given by $f(x) = (x - x_0)^4$, where $x_0 \in \mathbb{R}$ is a constant. Suppose that we apply Newton's method to the problem of minimizing f .

- a. Write down the update equation for Newton's method applied to the problem.

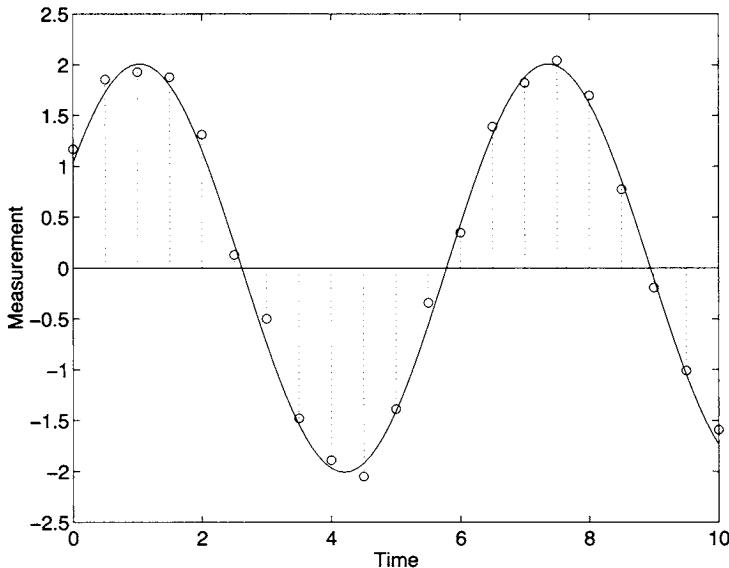


Figure 9.3 Sinusoid of best fit in Example 9.3.

- b. Let $y^{(k)} = |x^{(k)} - x_0|$, where $x^{(k)}$ is the k th iterate in Newton's method. Show that the sequence $\{y^{(k)}\}$ satisfies $y^{(k+1)} = \frac{2}{3}y^{(k)}$.
- c. Show that $x^{(k)} \rightarrow x_0$ for any initial guess $x^{(0)}$.
- d. Show that the order of convergence of the sequence $\{x^{(k)}\}$ in part b is 1.
- e. Theorem 9.1 states that under certain conditions, the order of convergence of Newton's method is at least 2. Why does that theorem not hold in this particular problem?

9.2 This question relates to the order of convergence of the secant method, using an argument similar to that of the proof of Theorem 9.1.

- a. Consider a function $f : \mathbb{R} \rightarrow \mathbb{R}$, $f \in \mathcal{C}^2$, such that x^* is a local minimizer and $f''(x^*) \neq 0$. Suppose that we apply the algorithm $x^{(k+1)} = x^{(k)} - \alpha_k f'(x^{(k)})$ such that $\{\alpha_k\}$ is a positive step-size sequence that converges to $1/f''(x^*)$. Show that if $x^{(k)} \rightarrow x^*$, then the order of convergence of the algorithm is *superlinear* (i.e., strictly greater than 1).
- b. Given part a, what can you say about the order of convergence of the secant algorithm?

9.3 Consider the problem of minimizing $f(x) = x^{\frac{4}{3}} = (\sqrt[3]{x})^4$, $x \in \mathbb{R}$. Note that 0 is the global minimizer of f .

- a. Write down the algorithm for Newton's method applied to this problem.
- b. Show that as long as the starting point is not 0, the algorithm in part a does not converge to 0 (no matter how close to 0 we start).

9.4 Consider *Rosenbrock's Function*: $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$, where $\mathbf{x} = [x_1, x_2]^\top$ (known to be a “nasty” function—often used as a benchmark for testing algorithms). This function is also known as the *banana function* because of the shape of its level sets.

- a. Prove that $[1, 1]^\top$ is the unique global minimizer of f over \mathbb{R}^2 .
- b. With a starting point of $[0, 0]^\top$, apply two iterations of Newton's method.
Hint:
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$
- c. Repeat part b using a gradient algorithm with a fixed step size of $\alpha_k = 0.05$ at each iteration.

9.5 Consider the modified Newton's algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)},$$

where $\alpha_k = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} - \alpha \mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)})$. Suppose that we apply the algorithm to a quadratic function $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} - \mathbf{x}^\top \mathbf{b}$, where $\mathbf{Q} = \mathbf{Q}^\top > 0$. Recall that the standard Newton's method reaches point \mathbf{x}^* such that $\nabla f(\mathbf{x}^*) = \mathbf{0}$ in just one step starting from any initial point $\mathbf{x}^{(0)}$. Does the modified Newton's algorithm above possess the same property?

CHAPTER 10

CONJUGATE DIRECTION METHODS

10.1 Introduction

The class of *conjugate direction methods* can be viewed as being intermediate between the method of steepest descent and Newton's method. The conjugate direction methods have the following properties:

1. Solve quadratics of n variables in n steps.
2. The usual implementation, the *conjugate gradient algorithm*, requires no Hessian matrix evaluations.
3. No matrix inversion and no storage of an $n \times n$ matrix are required.

The conjugate direction methods typically perform better than the method of steepest descent, but not as well as Newton's method. As we saw from the method of steepest descent and Newton's method, the crucial factor in the efficiency of an iterative search method is the direction of search at each iteration. For a quadratic function of n variables $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b}$, $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{Q} = \mathbf{Q}^\top > 0$, the best direction of search, as we shall see, is in the \mathbf{Q} -conjugate direction. Basically, two directions $\mathbf{d}^{(1)}$ and $\mathbf{d}^{(2)}$ in \mathbb{R}^n are

said to be \mathbf{Q} -conjugate if $\mathbf{d}^{(1)\top} \mathbf{Q} \mathbf{d}^{(2)} = 0$. In general, we have the following definition.

Definition 10.1 Let \mathbf{Q} be a real symmetric $n \times n$ matrix. The directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \dots, \mathbf{d}^{(m)}$ are \mathbf{Q} -conjugate if for all $i \neq j$, we have $\mathbf{d}^{(i)\top} \mathbf{Q} \mathbf{d}^{(j)} = 0$. ■

Lemma 10.1 Let \mathbf{Q} be a symmetric positive definite $n \times n$ matrix. If the directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(k)} \in \mathbb{R}^n$, $k \leq n - 1$, are nonzero and \mathbf{Q} -conjugate, then they are linearly independent. □

Proof. Let $\alpha_0, \dots, \alpha_k$ be scalars such that

$$\alpha_0 \mathbf{d}^{(0)} + \alpha_1 \mathbf{d}^{(1)} + \dots + \alpha_k \mathbf{d}^{(k)} = \mathbf{0}.$$

Premultiplying this equality by $\mathbf{d}^{(j)\top} \mathbf{Q}$, $0 \leq j \leq k$, yields

$$\alpha_j \mathbf{d}^{(j)\top} \mathbf{Q} \mathbf{d}^{(j)} = 0,$$

because all other terms $\mathbf{d}^{(j)\top} \mathbf{Q} \mathbf{d}^{(i)} = 0$, $i \neq j$, by \mathbf{Q} -conjugacy. But $\mathbf{Q} = \mathbf{Q}^\top > 0$ and $\mathbf{d}^{(j)} \neq \mathbf{0}$; hence $\alpha_j = 0$, $j = 0, 1, \dots, k$. Therefore, $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(k)}$, $k \leq n - 1$, are linearly independent. ■

Example 10.1 Let

$$\mathbf{Q} = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix}.$$

Note that $\mathbf{Q} = \mathbf{Q}^\top > 0$. The matrix \mathbf{Q} is positive definite because all its leading principal minors are positive:

$$\Delta_1 = 3 > 0, \quad \Delta_2 = \det \begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix} = 12 > 0, \quad \Delta_3 = \det \mathbf{Q} = 20 > 0.$$

Our goal is to construct a set of \mathbf{Q} -conjugate vectors $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \mathbf{d}^{(2)}$.

Let $\mathbf{d}^{(0)} = [1, 0, 0]^\top$, $\mathbf{d}^{(1)} = [d_1^{(1)}, d_2^{(1)}, d_3^{(1)}]^\top$, $\mathbf{d}^{(2)} = [d_1^{(2)}, d_2^{(2)}, d_3^{(2)}]^\top$. We require that $\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(1)} = 0$. We have

$$\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(1)} = [1, 0, 0] \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} d_1^{(1)} \\ d_2^{(1)} \\ d_3^{(1)} \end{bmatrix} = 3d_1^{(1)} + d_3^{(1)}.$$

Let $d_1^{(1)} = 1$, $d_2^{(1)} = 0$, $d_3^{(1)} = -3$. Then, $\mathbf{d}^{(1)} = [1, 0, -3]^\top$, and thus $\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(1)} = 0$.

To find the third vector $\mathbf{d}^{(2)}$, which would be \mathbf{Q} -conjugate with $\mathbf{d}^{(0)}$ and $\mathbf{d}^{(1)}$, we require that $\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(2)} = 0$ and $\mathbf{d}^{(1)\top} \mathbf{Q} \mathbf{d}^{(2)} = 0$. We have

$$\begin{aligned}\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(2)} &= 3d_1^{(2)} + d_3^{(2)} = 0, \\ \mathbf{d}^{(1)\top} \mathbf{Q} \mathbf{d}^{(2)} &= -6d_2^{(2)} - 8d_3^{(2)} = 0.\end{aligned}$$

If we take $\mathbf{d}^{(2)} = [1, 4, -3]^\top$, then the resulting set of vectors is mutually conjugate. ■

This method of finding \mathbf{Q} -conjugate vectors is inefficient. A systematic procedure for finding \mathbf{Q} -conjugate vectors can be devised using the idea underlying the *Gram-Schmidt process* of transforming a given basis of \mathbb{R}^n into an orthonormal basis of \mathbb{R}^n (see Exercise 10.1).

10.2 The Conjugate Direction Algorithm

We now present the conjugate direction algorithm for minimizing the quadratic function of n variables

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} - \mathbf{x}^\top \mathbf{b},$$

where $\mathbf{Q} = \mathbf{Q}^\top > 0, \mathbf{x} \in \mathbb{R}^n$. Note that because $\mathbf{Q} > 0$, the function f has a global minimizer that can be found by solving $\mathbf{Q}\mathbf{x} = \mathbf{b}$.

Basic Conjugate Direction Algorithm. Given a starting point $\mathbf{x}^{(0)}$ and \mathbf{Q} -conjugate directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(n-1)}$; for $k \geq 0$,

$$\begin{aligned}\mathbf{g}^{(k)} &= \nabla f(\mathbf{x}^{(k)}) = \mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b}, \\ \alpha_k &= -\frac{\mathbf{g}^{(k)\top} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(k)}}, \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}.\end{aligned}$$

Theorem 10.1 *For any starting point $\mathbf{x}^{(0)}$, the basic conjugate direction algorithm converges to the unique \mathbf{x}^* (that solves $\mathbf{Q}\mathbf{x} = \mathbf{b}$) in n steps; that is, $\mathbf{x}^{(n)} = \mathbf{x}^*$.* □

Proof. Consider $\mathbf{x}^* - \mathbf{x}^{(0)} \in \mathbb{R}^n$. Because the $\mathbf{d}^{(i)}$ are linearly independent, there exist constants β_i , $i = 0, \dots, n-1$, such that

$$\mathbf{x}^* - \mathbf{x}^{(0)} = \beta_0 \mathbf{d}^{(0)} + \dots + \beta_{n-1} \mathbf{d}^{(n-1)}.$$

Now premultiply both sides of this equation by $\mathbf{d}^{(k)\top} \mathbf{Q}$, $0 \leq k < n$, to obtain

$$\mathbf{d}^{(k)\top} \mathbf{Q} (\mathbf{x}^* - \mathbf{x}^{(0)}) = \beta_k \mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(k)},$$

where the terms $\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(i)} = 0$, $k \neq i$, by the \mathbf{Q} -conjugate property. Hence,

$$\beta_k = \frac{\mathbf{d}^{(k)\top} \mathbf{Q} (\mathbf{x}^* - \mathbf{x}^{(0)})}{\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(k)}}.$$

Now, we can write

$$\mathbf{x}^{(k)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)} + \cdots + \alpha_{k-1} \mathbf{d}^{(k-1)}.$$

Therefore,

$$\mathbf{x}^{(k)} - \mathbf{x}^{(0)} = \alpha_0 \mathbf{d}^{(0)} + \cdots + \alpha_{k-1} \mathbf{d}^{(k-1)}.$$

So writing

$$\mathbf{x}^* - \mathbf{x}^{(0)} = (\mathbf{x}^* - \mathbf{x}^{(k)}) + (\mathbf{x}^{(k)} - \mathbf{x}^{(0)})$$

and premultiplying the above by $\mathbf{d}^{(k)\top} \mathbf{Q}$, we obtain

$$\mathbf{d}^{(k)\top} \mathbf{Q} (\mathbf{x}^* - \mathbf{x}^{(0)}) = \mathbf{d}^{(k)\top} \mathbf{Q} (\mathbf{x}^* - \mathbf{x}^{(k)}) = -\mathbf{d}^{(k)\top} \mathbf{g}^{(k)},$$

because $\mathbf{g}^{(k)} = \mathbf{Q} \mathbf{x}^{(k)} - \mathbf{b}$ and $\mathbf{Q} \mathbf{x}^* = \mathbf{b}$. Thus,

$$\beta_k = -\frac{\mathbf{d}^{(k)\top} \mathbf{g}^{(k)}}{\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(k)}} = \alpha_k$$

and $\mathbf{x}^* = \mathbf{x}^{(n)}$, which completes the proof. ■

Example 10.2 Find the minimizer of

$$f(x_1, x_2) = \frac{1}{2} \mathbf{x}^\top \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \mathbf{x} - \mathbf{x}^\top \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \mathbf{x} \in \mathbb{R}^2,$$

using the conjugate direction method with the initial point $\mathbf{x}^{(0)} = [0, 0]^\top$, and \mathbf{Q} -conjugate directions $\mathbf{d}^{(0)} = [1, 0]^\top$ and $\mathbf{d}^{(1)} = [-\frac{3}{8}, \frac{3}{4}]^\top$.

We have

$$\mathbf{g}^{(0)} = -\mathbf{b} = [1, -1]^\top,$$

and hence

$$\alpha_0 = -\frac{\mathbf{g}^{(0)\top} \mathbf{d}^{(0)}}{\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(0)}} = -\frac{[1, -1]^\top \begin{bmatrix} 1 \\ 0 \end{bmatrix}}{[1, 0]^\top \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}} = -\frac{1}{4}.$$

Thus,

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -\frac{1}{4} \\ 0 \end{bmatrix}.$$

To find $\mathbf{x}^{(2)}$, we compute

$$\mathbf{g}^{(1)} = \mathbf{Q}\mathbf{x}^{(1)} - \mathbf{b} = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} -\frac{1}{4} \\ 0 \end{bmatrix} - \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{3}{2} \end{bmatrix}$$

and

$$\alpha_1 = -\frac{\mathbf{g}^{(1)\top} \mathbf{d}^{(1)}}{\mathbf{d}^{(1)\top} \mathbf{Q} \mathbf{d}^{(1)}} = -\frac{\begin{bmatrix} 0, -\frac{3}{2} \end{bmatrix} \begin{bmatrix} -\frac{3}{8} \\ \frac{3}{4} \end{bmatrix}}{\begin{bmatrix} -\frac{3}{8}, \frac{3}{4} \end{bmatrix} \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} -\frac{3}{8} \\ \frac{3}{4} \end{bmatrix}} = 2.$$

Therefore,

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha_1 \mathbf{d}^{(1)} = \begin{bmatrix} -\frac{1}{4} \\ 0 \end{bmatrix} + 2 \begin{bmatrix} -\frac{3}{8} \\ \frac{3}{4} \end{bmatrix} = \begin{bmatrix} -1 \\ \frac{3}{2} \end{bmatrix}.$$

Because f is a quadratic function in two variables, $\mathbf{x}^{(2)} = \mathbf{x}^*$. ■

For a quadratic function of n variables, the conjugate direction method reaches the solution after n steps. As we shall see below, the method also possesses a certain desirable property in the intermediate steps. To see this, suppose that we start at $\mathbf{x}^{(0)}$ and search in the direction $\mathbf{d}^{(0)}$ to obtain

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \left(\frac{\mathbf{g}^{(0)\top} \mathbf{d}^{(0)}}{\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(0)}} \right) \mathbf{d}^{(0)}.$$

We claim that

$$\mathbf{g}^{(1)\top} \mathbf{d}^{(0)} = 0.$$

To see this,

$$\begin{aligned} \mathbf{g}^{(1)\top} \mathbf{d}^{(0)} &= (\mathbf{Q}\mathbf{x}^{(1)} - \mathbf{b})^\top \mathbf{d}^{(0)} \\ &= \mathbf{x}^{(0)\top} \mathbf{Q} \mathbf{d}^{(0)} - \left(\frac{\mathbf{g}^{(0)\top} \mathbf{d}^{(0)}}{\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(0)}} \right) \mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(0)} - \mathbf{b}^\top \mathbf{d}^{(0)} \\ &= \mathbf{g}^{(0)\top} \mathbf{d}^{(0)} - \mathbf{g}^{(0)\top} \mathbf{d}^{(0)} = 0. \end{aligned}$$

The equation $\mathbf{g}^{(1)\top} \mathbf{d}^{(0)} = 0$ implies that α_0 has the property that $\alpha_0 = \arg \min \phi_0(\alpha)$, where $\phi_0(\alpha) = f(\mathbf{x}^{(0)} + \alpha \mathbf{d}^{(0)})$. To see this, apply the chain rule to get

$$\frac{d\phi_0}{d\alpha}(\alpha) = \nabla f(\mathbf{x}^{(0)} + \alpha \mathbf{d}^{(0)})^\top \mathbf{d}^{(0)}.$$

Evaluating the above at $\alpha = \alpha_0$, we get

$$\frac{d\phi_0}{d\alpha}(\alpha_0) = \mathbf{g}^{(1)\top} \mathbf{d}^{(0)} = 0.$$

Because ϕ_0 is a quadratic function of α , and the coefficient of the α^2 term in ϕ_0 is $\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(0)} > 0$, the above implies that $\alpha_0 = \arg \min_{\alpha \in \mathbb{R}} \phi_0(\alpha)$.

Using a similar argument, we can show that for all k ,

$$\mathbf{g}^{(k+1)\top} \mathbf{d}^{(k)} = 0$$

and hence

$$\alpha_k = \arg \min f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}).$$

In fact, an even stronger condition holds, as given by the following lemma.

Lemma 10.2 *In the conjugate direction algorithm,*

$$\mathbf{g}^{(k+1)\top} \mathbf{d}^{(i)} = 0$$

for all k , $0 \leq k \leq n - 1$, and $0 \leq i \leq k$. \square

Proof. Note that

$$\mathbf{Q}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = \mathbf{Q}\mathbf{x}^{(k+1)} - \mathbf{b} - (\mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b}) = \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)},$$

because $\mathbf{g}^{(k)} = \mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b}$. Thus,

$$\mathbf{g}^{(k+1)} = \mathbf{g}^{(k)} + \alpha_k \mathbf{Q} \mathbf{d}^{(k)}.$$

We prove the lemma by induction. The result is true for $k = 0$ because $\mathbf{g}^{(1)\top} \mathbf{d}^{(0)} = 0$, as shown before. We now show that if the result is true for $k - 1$ (i.e., $\mathbf{g}^{(k)\top} \mathbf{d}^{(i)} = 0$, $i \leq k - 1$), then it is true for k (i.e., $\mathbf{g}^{(k+1)\top} \mathbf{d}^{(i)} = 0$, $i \leq k$). Fix $k > 0$ and $0 \leq i < k$. By the induction hypothesis, $\mathbf{g}^{(k)\top} \mathbf{d}^{(i)} = 0$. Because

$$\mathbf{g}^{(k+1)} = \mathbf{g}^{(k)} + \alpha_k \mathbf{Q} \mathbf{d}^{(k)},$$

and $\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(i)} = 0$ by \mathbf{Q} -conjugacy, we have

$$\mathbf{g}^{(k+1)\top} \mathbf{d}^{(i)} = \mathbf{g}^{(k)\top} \mathbf{d}^{(i)} + \alpha_k \mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(i)} = 0.$$

It remains to be shown that

$$\mathbf{g}^{(k+1)\top} \mathbf{d}^{(k)} = 0.$$

Indeed,

$$\begin{aligned} \mathbf{g}^{(k+1)\top} \mathbf{d}^{(k)} &= (\mathbf{Q}\mathbf{x}^{(k+1)} - \mathbf{b})^\top \mathbf{d}^{(k)} \\ &= \left(\mathbf{x}^{(k)} - \frac{\mathbf{g}^{(k)\top} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(k)}} \mathbf{d}^{(k)} \right)^\top \mathbf{Q} \mathbf{d}^{(k)} - \mathbf{b}^\top \mathbf{d}^{(k)} \\ &= (\mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b})^\top \mathbf{d}^{(k)} - \mathbf{g}^{(k)\top} \mathbf{d}^{(k)} \\ &= 0, \end{aligned}$$

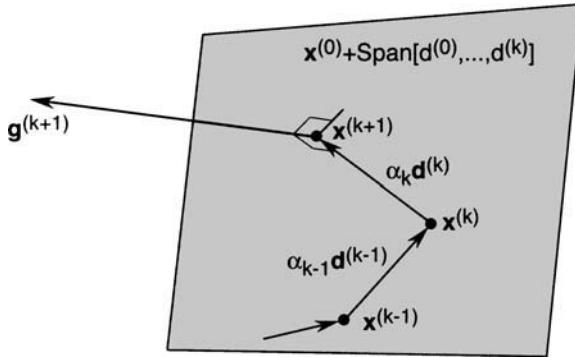


Figure 10.1 Illustration of Lemma 10.2.

because $\mathbf{Q}\mathbf{x}^{(k)} - \mathbf{b} = \mathbf{g}^{(k)}$.

Therefore, by induction, for all $0 \leq k \leq n - 1$ and $0 \leq i \leq k$,

$$\mathbf{g}^{(k+1)\top} \mathbf{d}^{(i)} = 0.$$

■

By Lemma 10.2 we see that $\mathbf{g}^{(k+1)}$ is orthogonal to any vector from the subspace spanned by $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(k)}$. Figure 10.1 illustrates this statement.

The lemma can be used to show an interesting optimal property of the conjugate direction algorithm. Specifically, we now show that not only does $f(\mathbf{x}^{(k+1)})$ satisfy $f(\mathbf{x}^{(k+1)}) = \min_{\alpha} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$, as indicated before, but also

$$f(\mathbf{x}^{(k+1)}) = \min_{a_0, \dots, a_k} f\left(\mathbf{x}^{(0)} + \sum_{i=0}^k a_i \mathbf{d}^{(i)}\right).$$

In other words, if we write

$$\mathcal{V}_k = \mathbf{x}^{(0)} + \text{span}[\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(k)}],$$

then we can express $f(\mathbf{x}^{(k+1)}) = \min_{\mathbf{x} \in \mathcal{V}_k} f(\mathbf{x})$. As k increases, the subspace $\text{span}[\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(k)}]$ “expands,” and will eventually fill the whole of \mathbb{R}^n (provided that the vectors $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots$ are linearly independent). Therefore, for some sufficiently large k , \mathbf{x}^* will lie in \mathcal{V}_k . For this reason, the above result is sometimes called the *expanding subspace theorem* (see, e.g., [88, p. 266]).

To prove the expanding subspace theorem, define the matrix $\mathbf{D}^{(k)}$ by

$$\mathbf{D}^{(k)} = [\mathbf{d}^{(0)}, \dots, \mathbf{d}^{(k)}];$$

that is, $\mathbf{d}^{(i)}$ is the i th column of $\mathbf{D}^{(k)}$. Note that $\mathbf{x}^{(0)} + \mathcal{R}(\mathbf{D}^{(k)}) = \mathcal{V}_k$. Also,

$$\begin{aligned}\mathbf{x}^{(k+1)} &= \mathbf{x}^{(0)} + \sum_{i=0}^k \alpha_i \mathbf{d}^{(i)} \\ &= \mathbf{x}^{(0)} + \mathbf{D}^{(k)} \boldsymbol{\alpha},\end{aligned}$$

where $\boldsymbol{\alpha} = [\alpha_0, \dots, \alpha_k]^\top$. Hence,

$$\mathbf{x}^{(k+1)} \in \mathbf{x}^{(0)} + \mathcal{R}(\mathbf{D}^{(k)}) = \mathcal{V}_k.$$

Now, consider any vector $\mathbf{x} \in \mathcal{V}_k$. There exists a vector \mathbf{a} such that $\mathbf{x} = \mathbf{x}^{(0)} + \mathbf{D}^{(k)} \mathbf{a}$. Let $\phi_k(\mathbf{a}) = f(\mathbf{x}^{(0)} + \mathbf{D}^{(k)} \mathbf{a})$. Note that ϕ_k is a quadratic function and has a unique minimizer that satisfies the FONC (see Exercises 6.33 and 10.7). By the chain rule,

$$D\phi_k(\mathbf{a}) = \nabla f(\mathbf{x}^{(0)} + \mathbf{D}^{(k)} \mathbf{a})^\top \mathbf{D}^{(k)}.$$

Therefore,

$$\begin{aligned}D\phi_k(\boldsymbol{\alpha}) &= \nabla f(\mathbf{x}^{(0)} + \mathbf{D}^{(k)} \boldsymbol{\alpha})^\top \mathbf{D}^{(k)} \\ &= \nabla f(\mathbf{x}^{(k+1)})^\top \mathbf{D}^{(k)} \\ &= \mathbf{g}^{(k+1)^\top} \mathbf{D}^{(k)}.\end{aligned}$$

By Lemma 10.2, $\mathbf{g}^{(k+1)^\top} \mathbf{D}^{(k)} = \mathbf{0}^\top$. Therefore, $\boldsymbol{\alpha}$ satisfies the FONC for the quadratic function ϕ_k , and hence $\boldsymbol{\alpha}$ is the minimizer of ϕ_k ; that is,

$$f(\mathbf{x}^{(k+1)}) = \min_{\mathbf{a}} f(\mathbf{x}^{(0)} + \mathbf{D}^{(k)} \mathbf{a}) = \min_{\mathbf{x} \in \mathcal{V}_k} f(\mathbf{x}),$$

which completes the proof of our result.

The conjugate direction algorithm is very effective. However, to use the algorithm, we need to specify the \mathbf{Q} -conjugate directions. Fortunately, there is a way to generate \mathbf{Q} -conjugate directions as we perform iterations. In the next section we discuss an algorithm that incorporates the generation of \mathbf{Q} -conjugate directions.

10.3 The Conjugate Gradient Algorithm

The conjugate gradient algorithm does not use prespecified conjugate directions, but instead computes the directions as the algorithm progresses. At each stage of the algorithm, the direction is calculated as a linear combination of the previous direction and the current gradient, in such a way that all the directions are mutually \mathbf{Q} -conjugate—hence the name *conjugate gradient algorithm*. This calculation exploits the fact that for a quadratic function of

n variables, we can locate the function minimizer by performing n searches along mutually conjugate directions.

As before, we consider the quadratic function

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{x} \in \mathbb{R}^n,$$

where $\mathbf{Q} = \mathbf{Q}^\top > 0$. Our first search direction from an initial point $\mathbf{x}^{(0)}$ is in the direction of steepest descent; that is,

$$\mathbf{d}^{(0)} = -\mathbf{g}^{(0)}.$$

Thus,

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)},$$

where

$$\alpha_0 = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(0)} + \alpha \mathbf{d}^{(0)}) = -\frac{\mathbf{g}^{(0)\top} \mathbf{d}^{(0)}}{\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(0)}}.$$

In the next stage, we search in a direction $\mathbf{d}^{(1)}$ that is \mathbf{Q} -conjugate to $\mathbf{d}^{(0)}$. We choose $\mathbf{d}^{(1)}$ as a linear combination of $\mathbf{g}^{(1)}$ and $\mathbf{d}^{(0)}$. In general, at the $(k+1)$ th step, we choose $\mathbf{d}^{(k+1)}$ to be a linear combination of $\mathbf{g}^{(k+1)}$ and $\mathbf{d}^{(k)}$. Specifically, we choose

$$\mathbf{d}^{(k+1)} = -\mathbf{g}^{(k+1)} + \beta_k \mathbf{d}^{(k)}, \quad k = 0, 1, 2, \dots$$

The coefficients β_k , $k = 1, 2, \dots$, are chosen in such a way that $\mathbf{d}^{(k+1)}$ is \mathbf{Q} -conjugate to $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(k)}$. This is accomplished by choosing β_k to be

$$\beta_k = \frac{\mathbf{g}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(k)}}.$$

The conjugate gradient algorithm is summarized below.

1. Set $k := 0$; select the initial point $\mathbf{x}^{(0)}$.
2. $\mathbf{g}^{(0)} = \nabla f(\mathbf{x}^{(0)})$. If $\mathbf{g}^{(0)} = \mathbf{0}$, stop; else, set $\mathbf{d}^{(0)} = -\mathbf{g}^{(0)}$.
3. $\alpha_k = -\frac{\mathbf{g}^{(k)\top} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(k)}}$.
4. $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$.
5. $\mathbf{g}^{(k+1)} = \nabla f(\mathbf{x}^{(k+1)})$. If $\mathbf{g}^{(k+1)} = \mathbf{0}$, stop.
6. $\beta_k = \frac{\mathbf{g}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(k)}}$.
7. $\mathbf{d}^{(k+1)} = -\mathbf{g}^{(k+1)} + \beta_k \mathbf{d}^{(k)}$.
8. Set $k := k + 1$; go to step 3.

Proposition 10.1 *In the conjugate gradient algorithm, the directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(n-1)}$ are \mathbf{Q} -conjugate.* \square

Proof. We use induction. We first show that $\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(1)} = 0$. To this end we write

$$\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(1)} = \mathbf{d}^{(0)\top} \mathbf{Q}(-\mathbf{g}^{(1)} + \beta_0 \mathbf{d}^{(0)}).$$

Substituting for

$$\beta_0 = \frac{\mathbf{g}^{(1)\top} \mathbf{Q} \mathbf{d}^{(0)}}{\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(0)}}$$

in the equation above, we see that $\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(1)} = 0$.

We now assume that $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(k)}$, $k < n - 1$, are \mathbf{Q} -conjugate directions. From Lemma 10.2 we have $\mathbf{g}^{(k+1)\top} \mathbf{d}^{(j)} = 0$, $j = 0, 1, \dots, k$. Thus, $\mathbf{g}^{(k+1)}$ is orthogonal to each of the directions $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(k)}$. We now show that

$$\mathbf{g}^{(k+1)\top} \mathbf{g}^{(j)} = 0, \quad j = 0, 1, \dots, k.$$

Fix $j \in \{0, \dots, k\}$. We have

$$\mathbf{d}^{(j)} = -\mathbf{g}^{(j)} + \beta_{j-1} \mathbf{d}^{(j-1)}.$$

Substituting this equation into the previous one yields

$$\mathbf{g}^{(k+1)\top} \mathbf{d}^{(j)} = 0 = -\mathbf{g}^{(k+1)\top} \mathbf{g}^{(j)} + \beta_{j-1} \mathbf{g}^{(k+1)\top} \mathbf{d}^{(j-1)}.$$

Because $\mathbf{g}^{(k+1)\top} \mathbf{d}^{(j-1)} = 0$, it follows that $\mathbf{g}^{(k+1)\top} \mathbf{g}^{(j)} = 0$.

We are now ready to show that $\mathbf{d}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(j)} = 0$, $j = 0, \dots, k$. We have

$$\mathbf{d}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(j)} = (-\mathbf{g}^{(k+1)} + \beta_k \mathbf{d}^{(k)})^\top \mathbf{Q} \mathbf{d}^{(j)}.$$

If $j < k$, then $\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(j)} = 0$, by virtue of the induction hypothesis. Hence, we have

$$\mathbf{d}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(j)} = -\mathbf{g}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(j)}.$$

But $\mathbf{g}^{(j+1)} = \mathbf{g}^{(j)} + \alpha_j \mathbf{Q} \mathbf{d}^{(j)}$. Because $\mathbf{g}^{(k+1)\top} \mathbf{g}^{(i)} = 0$, $i = 0, \dots, k$,

$$\mathbf{d}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(j)} = -\mathbf{g}^{(k+1)\top} \frac{(\mathbf{g}^{(j+1)} - \mathbf{g}^{(j)})}{\alpha_j} = 0.$$

Thus,

$$\mathbf{d}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(j)} = 0, \quad j = 0, \dots, k-1.$$

It remains to be shown that $\mathbf{d}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(k)} = 0$. We have

$$\mathbf{d}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(k)} = (-\mathbf{g}^{(k+1)} + \beta_k \mathbf{d}^{(k)})^\top \mathbf{Q} \mathbf{d}^{(k)}.$$

Using the expression for β_k , we get $\mathbf{d}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(k)} = 0$, which completes the proof. \blacksquare

Example 10.3 Consider the quadratic function

$$f(x_1, x_2, x_3) = \frac{3}{2}x_1^2 + 2x_2^2 + \frac{3}{2}x_3^2 + x_1x_3 + 2x_2x_3 - 3x_1 - x_3.$$

We find the minimizer using the conjugate gradient algorithm, using the starting point $\mathbf{x}^{(0)} = [0, 0, 0]^\top$.

We can represent f as

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b},$$

where

$$\mathbf{Q} = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix}.$$

We have

$$\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x}) = \mathbf{Q}\mathbf{x} - \mathbf{b} = [3x_1 + x_3 - 3, 4x_2 + 2x_3, x_1 + 2x_2 + 3x_3 - 1]^\top.$$

Hence,

$$\begin{aligned} \mathbf{g}^{(0)} &= [-3, 0, -1]^\top, \\ \mathbf{d}^{(0)} &= -\mathbf{g}^{(0)}, \\ \alpha_0 &= -\frac{\mathbf{g}^{(0)\top} \mathbf{d}^{(0)}}{\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(0)}} = \frac{10}{36} = 0.2778 \end{aligned}$$

and

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)} = [0.8333, 0, 0.2778]^\top.$$

The next stage yields

$$\begin{aligned} \mathbf{g}^{(1)} &= \nabla f(\mathbf{x}^{(1)}) = [-0.2222, 0.5556, 0.6667]^\top, \\ \beta_0 &= \frac{\mathbf{g}^{(1)\top} \mathbf{Q} \mathbf{d}^{(0)}}{\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(0)}} = 0.08025. \end{aligned}$$

We can now compute

$$\mathbf{d}^{(1)} = -\mathbf{g}^{(1)} + \beta_0 \mathbf{d}^{(0)} = [0.4630, -0.5556, -0.5864]^\top.$$

Hence,

$$\alpha_1 = -\frac{\mathbf{g}^{(1)\top} \mathbf{d}^{(1)}}{\mathbf{d}^{(1)\top} \mathbf{Q} \mathbf{d}^{(1)}} = 0.2187$$

and

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha_1 \mathbf{d}^{(1)} = [0.9346, -0.1215, 0.1495]^\top.$$

To perform the third iteration, we compute

$$\begin{aligned}\mathbf{g}^{(2)} &= \nabla f(\mathbf{x}^{(2)}) = [-0.04673, -0.1869, 0.1402]^\top, \\ \beta_1 &= \frac{\mathbf{g}^{(2)\top} \mathbf{Q} \mathbf{d}^{(1)}}{\mathbf{d}^{(1)\top} \mathbf{Q} \mathbf{d}^{(1)}} = 0.07075, \\ \mathbf{d}^{(2)} &= -\mathbf{g}^{(2)} + \beta_1 \mathbf{d}^{(1)} = [0.07948, 0.1476, -0.1817]^\top.\end{aligned}$$

Hence,

$$\alpha_2 = -\frac{\mathbf{g}^{(2)\top} \mathbf{d}^{(2)}}{\mathbf{d}^{(2)\top} \mathbf{Q} \mathbf{d}^{(2)}} = 0.8231$$

and

$$\mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \alpha_2 \mathbf{d}^{(2)} = [1.000, 0.000, 0.000]^\top.$$

Note that

$$\mathbf{g}^{(3)} = \nabla f(\mathbf{x}^{(3)}) = \mathbf{0},$$

as expected, because f is a quadratic function of three variables. Hence, $\mathbf{x}^* = \mathbf{x}^{(3)}$. ■

10.4 The Conjugate Gradient Algorithm for Nonquadratic Problems

In Section 10.3, we showed that the conjugate gradient algorithm is a conjugate direction method, and therefore minimizes a positive definite quadratic function of n variables in n steps. The algorithm can be extended to general nonlinear functions by interpreting $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} - \mathbf{x}^\top \mathbf{b}$ as a second-order Taylor series approximation of the objective function. Near the solution such functions behave approximately as quadratics, as suggested by the Taylor series expansion. For a quadratic, the matrix \mathbf{Q} , the Hessian of the quadratic, is constant. However, for a general nonlinear function the Hessian is a matrix that has to be reevaluated at each iteration of the algorithm. This can be computationally very expensive. Thus, an efficient implementation of the conjugate gradient algorithm that eliminates the Hessian evaluation at each step is desirable.

Observe that \mathbf{Q} appears only in the computation of the scalars α_k and β_k . Because

$$\alpha_k = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}),$$

the closed-form formula for α_k in the algorithm can be replaced by a numerical line search procedure. Therefore, we need only concern ourselves with the formula for β_k . Fortunately, elimination of \mathbf{Q} from the formula is possible and results in algorithms that depend only on the function and gradient values at

each iteration. We now discuss modifications of the conjugate gradient algorithm for a quadratic function for the case in which the Hessian is unknown but in which objective function values and gradients are available. The modifications are all based on algebraically manipulating the formula β_k in such a way that \mathbf{Q} is eliminated. We discuss three well-known modifications.

Hestenes-Stiefel Formula. Recall that

$$\beta_k = \frac{\mathbf{g}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(k)}}{\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(k)}}.$$

The Hestenes-Stiefel formula is based on replacing the term $\mathbf{Q} \mathbf{d}^{(k)}$ by the term $(\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)})/\alpha_k$. The two terms are equal in the quadratic case, as we now show. Now, $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$. Premultiplying both sides by \mathbf{Q} , subtracting \mathbf{b} from both sides, and recognizing that $\mathbf{g}^{(k)} = \mathbf{Q} \mathbf{x}^{(k)} - \mathbf{b}$, we get $\mathbf{g}^{(k+1)} = \mathbf{g}^{(k)} + \alpha_k \mathbf{Q} \mathbf{d}^{(k)}$, which we can rewrite as $\mathbf{Q} \mathbf{d}^{(k)} = (\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)})/\alpha_k$. Substituting this into the original equation for β_k gives the *Hestenes-Stiefel formula*

$$\beta_k = \frac{\mathbf{g}^{(k+1)\top} [\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}{\mathbf{d}^{(k)\top} [\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}.$$

Polak-Ribière Formula. Starting from the Hestenes-Stiefel formula, we multiply out the denominator to get

$$\beta_k = \frac{\mathbf{g}^{(k+1)\top} [\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}{\mathbf{d}^{(k)\top} \mathbf{g}^{(k+1)} - \mathbf{d}^{(k)\top} \mathbf{g}^{(k)}}.$$

By Lemma 10.2, $\mathbf{d}^{(k)\top} \mathbf{g}^{(k+1)} = 0$. Also, since $\mathbf{d}^{(k)} = -\mathbf{g}^{(k)} + \beta_{k-1} \mathbf{d}^{(k-1)}$, and premultiplying this by $\mathbf{g}^{(k)\top}$, we get

$$\mathbf{g}^{(k)\top} \mathbf{d}^{(k)} = -\mathbf{g}^{(k)\top} \mathbf{g}^{(k)} + \beta_{k-1} \mathbf{g}^{(k)\top} \mathbf{d}^{(k-1)} = -\mathbf{g}^{(k)\top} \mathbf{g}^{(k)},$$

where once again we used Lemma 10.2. Hence, we get the Polak-Ribière formula

$$\beta_k = \frac{\mathbf{g}^{(k+1)\top} [\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}{\mathbf{g}^{(k)\top} \mathbf{g}^{(k)}}.$$

Fletcher-Reeves Formula. Starting with the Polak-Ribière formula, we multiply out the numerator to get

$$\beta_k = \frac{\mathbf{g}^{(k+1)\top} \mathbf{g}^{(k+1)} - \mathbf{g}^{(k+1)\top} \mathbf{g}^{(k)}}{\mathbf{g}^{(k)\top} \mathbf{g}^{(k)}}.$$

We now use the fact that $\mathbf{g}^{(k+1)\top} \mathbf{g}^{(k)} = 0$, which we get by using the equation

$$\mathbf{g}^{(k+1)\top} \mathbf{d}^{(k)} = -\mathbf{g}^{(k+1)\top} \mathbf{g}^{(k)} + \beta_{k-1} \mathbf{g}^{(k+1)\top} \mathbf{d}^{(k-1)}$$

and applying Lemma 10.2. This leads to the Fletcher-Reeves formula

$$\beta_k = \frac{\mathbf{g}^{(k+1)\top} \mathbf{g}^{(k+1)}}{\mathbf{g}^{(k)\top} \mathbf{g}^{(k)}}.$$

The formulas above give us conjugate gradient algorithms that do not require explicit knowledge of the Hessian matrix \mathbf{Q} . All we need are the objective function and gradient values at each iteration. For the quadratic case the three expressions for β_k are exactly equal. However, this is not the case for a general nonlinear objective function.

We need a few more slight modifications to apply the algorithm to general nonlinear functions in practice. First, as mentioned in our discussion of the steepest descent algorithm (Section 8.2), the stopping criterion $\nabla f(\mathbf{x}^{(k+1)}) = \mathbf{0}$ is not practical. A suitable practical stopping criterion, such as those discussed in Section 8.2, needs to be used.

For nonquadratic problems, the algorithm will not usually converge in n steps, and as the algorithm progresses, the “ \mathbf{Q} -conjugacy” of the direction vectors will tend to deteriorate. Thus, a common practice is to reinitialize the direction vector to the negative gradient after every few iterations (e.g., n or $n + 1$) and continue until the algorithm satisfies the stopping criterion.

A very important issue in minimization problems of nonquadratic functions is the line search. The purpose of the line search is to minimize $\phi_k(\alpha) = f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$ with respect to $\alpha \geq 0$. A typical approach is to bracket or box in the minimizer and then estimate it. The accuracy of the line search is a critical factor in the performance of the conjugate gradient algorithm. If the line search is known to be inaccurate, the Hestenes-Stiefel formula for β_k is recommended [69].

In general, the choice of which formula for β_k to use depends on the objective function. For example, the Polak-Ribière formula is known to perform far better than the Fletcher-Reeves formula in some cases but not in others. In fact, there are cases in which the $\mathbf{g}^{(k)}$, $k = 1, 2, \dots$, are bounded away from zero when the Polak-Ribière formula is used (see [107]). In the study by Powell in [107], a global convergence analysis suggests that the Fletcher-Reeves formula for β_k is superior. Powell further suggests another formula for β_k :

$$\beta_k = \max \left\{ 0, \frac{\mathbf{g}^{(k+1)\top} [\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}]}{\mathbf{g}^{(k)\top} \mathbf{g}^{(k)}} \right\}.$$

For general results on the convergence of conjugate gradient methods, we refer the reader to [135]. For an application of conjugate gradient algorithms to Wiener filtering, see [116], [117], and [118].

Conjugate gradient algorithms are related to *Krylov subspace methods* (see Exercise 10.6). Krylov-subspace-iteration methods, initiated by Magnus Hestenes, Eduard Stiefel, and Cornelius Lanczos, have been declared one of the 10 algorithms with the greatest influence on the development and practice of science and engineering in the twentieth century [40].

For control perspective on the conjugate gradient algorithm, derived from a proportional-plus-derivative (PD) controller architecture, see [4]. In addition, these authors offer a control perspective on Krylov-subspace-iteration methods as discrete feedback control systems.

EXERCISES

10.1 (Adopted from [88, Exercise 9.8(1)]) Let \mathbf{Q} be a real symmetric positive definite $n \times n$ matrix. Given an arbitrary set of linearly independent vectors $\{\mathbf{p}^{(0)}, \dots, \mathbf{p}^{(n-1)}\}$ in \mathbb{R}^n , the *Gram-Schmidt procedure* generates a set of vectors $\{\mathbf{d}^{(0)}, \dots, \mathbf{d}^{(n-1)}\}$ as follows:

$$\begin{aligned}\mathbf{d}^{(0)} &= \mathbf{p}^{(0)}, \\ \mathbf{d}^{(k+1)} &= \mathbf{p}^{(k+1)} - \sum_{i=0}^k \frac{\mathbf{p}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(i)}}{\mathbf{d}^{(i)\top} \mathbf{Q} \mathbf{d}^{(i)}} \mathbf{d}^{(i)}.\end{aligned}$$

Show that the vectors $\mathbf{d}^{(0)}, \dots, \mathbf{d}^{(n-1)}$ are \mathbf{Q} -conjugate.

10.2 Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be the quadratic function

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} - \mathbf{x}^\top \mathbf{b},$$

where $\mathbf{Q} = \mathbf{Q}^\top > 0$. Given a set of directions $\{\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots\} \subset \mathbb{R}^n$, consider the algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},$$

where α_k is the step size. Suppose that $\mathbf{g}^{(k+1)\top} \mathbf{d}^{(i)} = 0$ for all $k = 0, \dots, n-1$ and $i = 0, \dots, k$, where $\mathbf{g}^{(k+1)} = \nabla f(\mathbf{x}^{(k+1)})$. Show that if $\mathbf{g}^{(k)\top} \mathbf{d}^{(k)} \neq 0$ for all $k = 0, \dots, n-1$, then $\mathbf{d}^{(0)}, \dots, \mathbf{d}^{(n-1)}$ are \mathbf{Q} -conjugate.

10.3 Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be given by $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} - \mathbf{x}^\top \mathbf{b}$, where $\mathbf{b} \in \mathbb{R}^n$ and \mathbf{Q} is a real symmetric positive definite $n \times n$ matrix. Show that in the conjugate gradient method for this f , $\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(k)} = -\mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{g}^{(k)}$.

10.4 Let \mathbf{Q} be a real $n \times n$ symmetric matrix.

- a. Show that there exists a \mathbf{Q} -conjugate set $\{\mathbf{d}^{(1)}, \dots, \mathbf{d}^{(n)}\}$ such that each $\mathbf{d}^{(i)}$ ($i = 1, \dots, n$) is an eigenvector of \mathbf{Q} .
Hint: Use the fact that for any real symmetric $n \times n$ matrix, there exists a set $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ of its eigenvectors such that $\mathbf{v}_i^\top \mathbf{v}_j = 0$ for all $i, j = 1, \dots, n$, $i \neq j$.
- b. Suppose that \mathbf{Q} is positive definite. Show that if $\{\mathbf{d}^{(1)}, \dots, \mathbf{d}^{(n)}\}$ is a \mathbf{Q} -conjugate set that is also orthogonal (i.e., $\mathbf{d}^{(i)\top} \mathbf{d}^{(j)} = 0$ for all $i, j = 1, \dots, n$, $i \neq j$), and $\mathbf{d}^{(i)} \neq \mathbf{0}$, $i = 1, \dots, n$, then each $\mathbf{d}^{(i)}$, $i = 1, \dots, n$, is an eigenvector of \mathbf{Q} .

10.5 Consider the following algorithm for minimizing a function f :

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},$$

where $\alpha_k = \arg \min_{\alpha} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$. Let $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$ (as usual).

Suppose that f is quadratic with Hessian \mathbf{Q} . We choose $\mathbf{d}^{(k+1)} = \gamma_k \mathbf{g}^{(k+1)} + \mathbf{d}^{(k)}$, and we wish the directions $\mathbf{d}^{(k)}$ and $\mathbf{d}^{(k+1)}$ to be \mathbf{Q} -conjugate. Find a formula for γ_k in terms of $\mathbf{d}^{(k)}$, $\mathbf{g}^{(k+1)}$, and \mathbf{Q} .

10.6 Consider the algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},$$

with $\alpha_k \in \mathbb{R}$ scalar and $\mathbf{x}^{(0)} = \mathbf{0}$, applied to the quadratic function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ given by

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} - \mathbf{b}^\top \mathbf{x},$$

where $\mathbf{Q} > 0$. As usual, write $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$. Suppose that the search directions are generated according to

$$\mathbf{d}^{(k+1)} = a_k \mathbf{g}^{(k+1)} + b_k \mathbf{d}^{(k)},$$

where a_k and b_k are real constants, and by convention we take $\mathbf{d}^{(-1)} = \mathbf{0}$.

- a. Define the subspace $\mathcal{V}_k = \text{span}[\mathbf{b}, \mathbf{Q}\mathbf{b}, \dots, \mathbf{Q}^{k-1}\mathbf{b}]$ (called the *Krylov subspace of order k*). Show that $\mathbf{d}^{(k)} \in \mathcal{V}_{k+1}$ and $\mathbf{x}^{(k)} \in \mathcal{V}_k$.

Hint: Use induction. Note that $\mathcal{V}_0 = \{\mathbf{0}\}$ and $\mathcal{V}_1 = \text{span}[\mathbf{b}]$.

- b. In light of part a, what can you say about the “optimality” of the conjugate gradient algorithm with respect to the Krylov subspace?

10.7 Consider the quadratic function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ given by

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} - \mathbf{x}^\top \mathbf{b},$$

where $\mathbf{Q} = \mathbf{Q}^\top > 0$. Let $\mathbf{D} \in \mathbb{R}^{n \times r}$ be of rank r and $\mathbf{x}_0 \in \mathbb{R}^n$. Define the function $\phi : \mathbb{R}^r \rightarrow \mathbb{R}$ by

$$\phi(\mathbf{a}) = f(\mathbf{x}_0 + \mathbf{D}\mathbf{a}).$$

Show that ϕ is a quadratic function with a positive definite quadratic term.

10.8 Consider a conjugate gradient algorithm applied to a quadratic function.

- a. Show that the gradients associated with the algorithm are mutually orthogonal. Specifically, show that $\mathbf{g}^{(k+1)\top} \mathbf{g}^{(i)} = 0$ for all $0 \leq k \leq n-1$ and $0 \leq i \leq k$.

Hint: Write $\mathbf{g}^{(i)}$ in terms of $\mathbf{d}^{(i)}$ and $\mathbf{d}^{(i-1)}$.

- b. Show that the gradients associated with the algorithm are \mathbf{Q} -conjugate if separated by at least two iterations. Specifically, show that $\mathbf{g}^{(k+1)\top} \mathbf{Q} \mathbf{g}^{(i)} = 0$ for all $0 \leq k \leq n-1$ and $0 \leq i \leq k-1$.

10.9 Represent the function

$$f(x_1, x_2) = \frac{5}{2}x_1^2 + x_2^2 - 3x_1x_2 - x_2 - 7$$

in the form $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b} + c$. Then use the *conjugate gradient algorithm* to construct a vector $\mathbf{d}^{(1)}$ that is \mathbf{Q} -conjugate with $\mathbf{d}^{(0)} = \nabla f(\mathbf{x}^{(0)})$, where $\mathbf{x}^{(0)} = \mathbf{0}$.

10.10 Let $f(\mathbf{x})$, $\mathbf{x} = [x_1, x_2]^\top \in \mathbb{R}^2$, be given by

$$f(\mathbf{x}) = \frac{5}{2}x_1^2 + \frac{1}{2}x_2^2 + 2x_1x_2 - 3x_1 - x_2.$$

- a. Express $f(\mathbf{x})$ in the form of $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b}$.
- b. Find the minimizer of f using the conjugate gradient algorithm. Use a starting point of $\mathbf{x}^{(0)} = [0, 0]^\top$.
- c. Calculate the minimizer of f analytically from \mathbf{Q} and \mathbf{b} , and check it with your answer in part b.

10.11 Write a MATLAB program to implement the conjugate gradient algorithm for general functions. Use the secant method for the line search (e.g., the MATLAB function of Exercise 7.11). Test the different formulas for β_k on Rosenbrock's function (see Exercise 9.4) with an initial condition $\mathbf{x}^{(0)} = [-2, 2]^\top$. For this exercise, reinitialize the update direction to the negative gradient every six iterations.

CHAPTER 11

QUASI-NEWTON METHODS

11.1 Introduction

Newton's method is one of the more successful algorithms for optimization. If it converges, it has a quadratic order of convergence. However, as pointed out before, for a general nonlinear objective function, convergence to a solution cannot be guaranteed from an arbitrary initial point $\mathbf{x}^{(0)}$. In general, if the initial point is not sufficiently close to the solution, then the algorithm may not possess the descent property [i.e., $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$ for some k].

Recall that the idea behind Newton's method is to locally approximate the function f being minimized, at every iteration, by a quadratic function. The minimizer for the quadratic approximation is used as the starting point for the next iteration. This leads to Newton's recursive algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)}.$$

We may try to guarantee that the algorithm has the descent property by modifying the original algorithm as follows:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)},$$

where α_k is chosen to ensure that

$$f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)}).$$

For example, we may choose $\alpha_k = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} - \alpha \mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)})$ (see Theorem 9.2). We can then determine an appropriate value of α_k by performing a line search in the direction $-\mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)}$. Note that although the line search is simply the minimization of the real variable function $\phi_k(\alpha) = f(\mathbf{x}^{(k)} - \alpha \mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)})$, it is not a trivial problem to solve.

A computational drawback of Newton's method is the need to evaluate $\mathbf{F}(\mathbf{x}^{(k)})$ and solve the equation $\mathbf{F}(\mathbf{x}^{(k)}) \mathbf{d}^{(k)} = -\mathbf{g}^{(k)}$ [i.e., compute $\mathbf{d}^{(k)} = -\mathbf{F}(\mathbf{x}^{(k)})^{-1} \mathbf{g}^{(k)}$]. To avoid the computation of $\mathbf{F}(\mathbf{x}^{(k)})^{-1}$, the quasi-Newton methods use an approximation to $\mathbf{F}(\mathbf{x}^{(k)})^{-1}$ in place of the true inverse. This approximation is updated at every stage so that it exhibits at least some properties of $\mathbf{F}(\mathbf{x}^{(k)})^{-1}$. To get some idea about the properties that an approximation to $\mathbf{F}(\mathbf{x}^{(k)})^{-1}$ should satisfy, consider the formula

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha \mathbf{H}_k \mathbf{g}^{(k)},$$

where \mathbf{H}_k is an $n \times n$ real matrix and α is a positive search parameter. Expanding f about $\mathbf{x}^{(k)}$ yields

$$\begin{aligned} f(\mathbf{x}^{(k+1)}) &= f(\mathbf{x}^{(k)}) + \mathbf{g}^{(k)\top} (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) + o(\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|) \\ &= f(\mathbf{x}^{(k)}) - \alpha \mathbf{g}^{(k)\top} \mathbf{H}_k \mathbf{g}^{(k)} + o(\|\mathbf{H}_k \mathbf{g}^{(k)}\| \alpha). \end{aligned}$$

As α tends to zero, the second term on the right-hand side of this equation dominates the third. Thus, to guarantee a decrease in f for small α , we have to have

$$\mathbf{g}^{(k)\top} \mathbf{H}_k \mathbf{g}^{(k)} > 0.$$

A simple way to ensure this is to require that \mathbf{H}_k be positive definite. We have proved the following result.

Proposition 11.1 *Let $f \in \mathcal{C}^1$, $\mathbf{x}^{(k)} \in \mathbb{R}^n$, $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)}) \neq \mathbf{0}$, and \mathbf{H}_k an $n \times n$ real symmetric positive definite matrix. If we set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{H}_k \mathbf{g}^{(k)}$, where $\alpha_k = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} - \alpha \mathbf{H}_k \mathbf{g}^{(k)})$, then $\alpha_k > 0$ and $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$.* \square

In constructing an approximation to the inverse of the Hessian matrix, we should use only the objective function and gradient values. Thus, if we can find a suitable method of choosing \mathbf{H}_k , the iteration may be carried out without any evaluation of the Hessian and without the solution of any set of linear equations.

11.2 Approximating the Inverse Hessian

Let $\mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2, \dots$ be successive approximations of the inverse $\mathbf{F}(\mathbf{x}^{(k)})^{-1}$ of the Hessian. We now derive a condition that the approximations should

satisfy, which forms the starting point for our subsequent discussion of quasi-Newton algorithms. To begin, suppose first that the Hessian matrix $\mathbf{F}(\mathbf{x})$ of the objective function f is constant and independent of \mathbf{x} . In other words, the objective function is quadratic, with Hessian $\mathbf{F}(\mathbf{x}) = \mathbf{Q}$ for all \mathbf{x} , where $\mathbf{Q} = \mathbf{Q}^\top$. Then,

$$\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)} = \mathbf{Q}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}).$$

Let

$$\Delta\mathbf{g}^{(k)} \triangleq \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}$$

and

$$\Delta\mathbf{x}^{(k)} \triangleq \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}.$$

Then, we may write

$$\Delta\mathbf{g}^{(k)} = \mathbf{Q}\Delta\mathbf{x}^{(k)}.$$

We start with a real symmetric positive definite matrix \mathbf{H}_0 . Note that given k , the matrix \mathbf{Q}^{-1} satisfies

$$\mathbf{Q}^{-1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, \quad 0 \leq i \leq k.$$

Therefore, we also impose the requirement that the approximation \mathbf{H}_{k+1} of the Hessian satisfy

$$\mathbf{H}_{k+1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}, \quad 0 \leq i \leq k.$$

If n steps are involved, then moving in n directions $\Delta\mathbf{x}^{(0)}, \Delta\mathbf{x}^{(1)}, \dots, \Delta\mathbf{x}^{(n-1)}$ yields

$$\begin{aligned} \mathbf{H}_n\Delta\mathbf{g}^{(0)} &= \Delta\mathbf{x}^{(0)}, \\ \mathbf{H}_n\Delta\mathbf{g}^{(1)} &= \Delta\mathbf{x}^{(1)}, \\ &\vdots \\ \mathbf{H}_n\Delta\mathbf{g}^{(n-1)} &= \Delta\mathbf{x}^{(n-1)}. \end{aligned}$$

This set of equations can be represented as

$$\mathbf{H}_n[\Delta\mathbf{g}^{(0)}, \Delta\mathbf{g}^{(1)}, \dots, \Delta\mathbf{g}^{(n-1)}] = [\Delta\mathbf{x}^{(0)}, \Delta\mathbf{x}^{(1)}, \dots, \Delta\mathbf{x}^{(n-1)}].$$

Note that \mathbf{Q} satisfies

$$\mathbf{Q}[\Delta\mathbf{x}^{(0)}, \Delta\mathbf{x}^{(1)}, \dots, \Delta\mathbf{x}^{(n-1)}] = [\Delta\mathbf{g}^{(0)}, \Delta\mathbf{g}^{(1)}, \dots, \Delta\mathbf{g}^{(n-1)}]$$

and

$$\mathbf{Q}^{-1}[\Delta\mathbf{g}^{(0)}, \Delta\mathbf{g}^{(1)}, \dots, \Delta\mathbf{g}^{(n-1)}] = [\Delta\mathbf{x}^{(0)}, \Delta\mathbf{x}^{(1)}, \dots, \Delta\mathbf{x}^{(n-1)}].$$

Therefore, if $[\Delta\mathbf{g}^{(0)}, \Delta\mathbf{g}^{(1)}, \dots, \Delta\mathbf{g}^{(n-1)}]$ is nonsingular, then \mathbf{Q}^{-1} is determined uniquely after n steps, via

$$\mathbf{Q}^{-1} = \mathbf{H}_n = [\Delta\mathbf{x}^{(0)}, \Delta\mathbf{x}^{(1)}, \dots, \Delta\mathbf{x}^{(n-1)}][\Delta\mathbf{g}^{(0)}, \Delta\mathbf{g}^{(1)}, \dots, \Delta\mathbf{g}^{(n-1)}]^{-1}.$$

As a consequence, we conclude that if \mathbf{H}_n satisfies the equations $\mathbf{H}_n \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}$, $0 \leq i \leq n - 1$, then the algorithm $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{H}_k \mathbf{g}^{(k)}$, $\alpha_k = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} - \alpha \mathbf{H}_k \mathbf{g}^{(k)})$, is guaranteed to solve problems with quadratic objective functions in $n + 1$ steps, because the update $\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \alpha_n \mathbf{H}_n \mathbf{g}^{(n)}$ is equivalent to Newton's algorithm. In fact, as we shall see below (Theorem 11.1), such algorithms solve quadratic problems of n variables in at most n steps.

The considerations above illustrate the basic idea behind the quasi-Newton methods. Specifically, quasi-Newton algorithms have the form

$$\begin{aligned}\mathbf{d}^{(k)} &= -\mathbf{H}_k \mathbf{g}^{(k)}, \\ \alpha_k &= \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}), \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},\end{aligned}$$

where the matrices $\mathbf{H}_0, \mathbf{H}_1, \dots$ are symmetric. In the quadratic case these matrices are required to satisfy

$$\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}, \quad 0 \leq i \leq k,$$

where $\Delta \mathbf{x}^{(i)} = \mathbf{x}^{(i+1)} - \mathbf{x}^{(i)} = \alpha_i \mathbf{d}^{(i)}$ and $\Delta \mathbf{g}^{(i)} = \mathbf{g}^{(i+1)} - \mathbf{g}^{(i)} = \mathbf{Q} \Delta \mathbf{x}^{(i)}$. It turns out that quasi-Newton methods are also conjugate direction methods, as stated in the following.

Theorem 11.1 *Consider a quasi-Newton algorithm applied to a quadratic function with Hessian $\mathbf{Q} = \mathbf{Q}^\top$ such that for $0 \leq k < n - 1$,*

$$\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}, \quad 0 \leq i \leq k,$$

where $\mathbf{H}_{k+1} = \mathbf{H}_{k+1}^\top$. If $\alpha_i \neq 0$, $0 \leq i \leq k$, then $\mathbf{d}^{(0)}, \dots, \mathbf{d}^{(k+1)}$ are \mathbf{Q} -conjugate. \square

Proof. We proceed by induction. We begin with the $k = 0$ case: that $\mathbf{d}^{(0)}$ and $\mathbf{d}^{(1)}$ are \mathbf{Q} -conjugate. Because $\alpha_0 \neq 0$, we can write $\mathbf{d}^{(0)} = \Delta \mathbf{x}^{(0)} / \alpha_0$. Hence,

$$\begin{aligned}\mathbf{d}^{(1)\top} \mathbf{Q} \mathbf{d}^{(0)} &= -\mathbf{g}^{(1)\top} \mathbf{H}_1 \mathbf{Q} \mathbf{d}^{(0)} \\ &= -\mathbf{g}^{(1)\top} \mathbf{H}_1 \frac{\mathbf{Q} \Delta \mathbf{x}^{(0)}}{\alpha_0} \\ &= -\mathbf{g}^{(1)\top} \frac{\mathbf{H}_1 \Delta \mathbf{g}^{(0)}}{\alpha_0} \\ &= -\mathbf{g}^{(1)\top} \frac{\Delta \mathbf{x}^{(0)}}{\alpha_0} \\ &= -\mathbf{g}^{(1)\top} \mathbf{d}^{(0)}.\end{aligned}$$

But $\mathbf{g}^{(1)\top} \mathbf{d}^{(0)} = 0$ as a consequence of $\alpha_0 > 0$ being the minimizer of $\phi(\alpha) = f(\mathbf{x}^{(0)} + \alpha \mathbf{d}^{(0)})$ (see Exercise 11.1). Hence, $\mathbf{d}^{(1)\top} \mathbf{Q} \mathbf{d}^{(0)} = 0$.

Assume that the result is true for $k - 1$ (where $k < n - 1$). We now prove the result for k , that is, that $\mathbf{d}^{(0)}, \dots, \mathbf{d}^{(k+1)}$ are \mathbf{Q} -conjugate. It suffices to show that $\mathbf{d}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(i)} = 0$, $0 \leq i \leq k$. Given i , $0 \leq i \leq k$, using the same algebraic steps as in the $k = 0$ case, and using the assumption that $\alpha_i \neq 0$, we obtain

$$\begin{aligned} \mathbf{d}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(i)} &= -\mathbf{g}^{(k+1)\top} \mathbf{H}_{k+1} \mathbf{Q} \mathbf{d}^{(i)} \\ &\vdots \\ &= -\mathbf{g}^{(k+1)\top} \mathbf{d}^{(i)}. \end{aligned}$$

Because $\mathbf{d}^{(0)}, \dots, \mathbf{d}^{(k)}$ are \mathbf{Q} -conjugate by assumption, we conclude from Lemma 10.2 that $\mathbf{g}^{(k+1)\top} \mathbf{d}^{(i)} = 0$. Hence, $\mathbf{d}^{(k+1)\top} \mathbf{Q} \mathbf{d}^{(i)} = 0$, which completes the proof. ■

By Theorem 11.1 we conclude that a quasi-Newton algorithm solves a quadratic of n variables in at most n steps.

Note that the equations that the matrices \mathbf{H}_k are required to satisfy do not determine those matrices uniquely. Thus, we have some freedom in the way we compute the \mathbf{H}_k . In the methods we describe, we compute \mathbf{H}_{k+1} by adding a correction to \mathbf{H}_k . In the following sections we consider three specific updating formulas.

11.3 The Rank One Correction Formula

In the *rank one correction formula*, the correction term is symmetric and has the form $a_k \mathbf{z}^{(k)} \mathbf{z}^{(k)\top}$, where $a_k \in \mathbb{R}$ and $\mathbf{z}^{(k)} \in \mathbb{R}^n$. Therefore, the update equation is

$$\mathbf{H}_{k+1} = \mathbf{H}_k + a_k \mathbf{z}^{(k)} \mathbf{z}^{(k)\top}.$$

Note that

$$\text{rank } \mathbf{z}^{(k)} \mathbf{z}^{(k)\top} = \text{rank} \left(\begin{bmatrix} z_1^{(k)} \\ \vdots \\ z_n^{(k)} \end{bmatrix} \begin{bmatrix} z_1^{(k)} & \cdots & z_n^{(k)} \end{bmatrix} \right) = 1$$

and hence the name *rank one correction* [it is also called the *single-rank symmetric (SRS) algorithm*]. The product $\mathbf{z}^{(k)} \mathbf{z}^{(k)\top}$ is sometimes referred to as the *dyadic product* or *outer product*. Observe that if \mathbf{H}_k is symmetric, then so is \mathbf{H}_{k+1} .

Our goal now is to determine a_k and $\mathbf{z}^{(k)}$, given \mathbf{H}_k , $\Delta \mathbf{g}^{(k)}$, $\Delta \mathbf{x}^{(k)}$, so that the required relationship discussed in Section 11.2 is satisfied; namely,

$\mathbf{H}_{k+1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}$, $i = 1, \dots, k$. To begin, let us first consider the condition $\mathbf{H}_{k+1}\Delta\mathbf{g}^{(k)} = \Delta\mathbf{x}^{(k)}$. In other words, given \mathbf{H}_k , $\Delta\mathbf{g}^{(k)}$, and $\Delta\mathbf{x}^{(k)}$, we wish to find a_k and $\mathbf{z}^{(k)}$ to ensure that

$$\mathbf{H}_{k+1}\Delta\mathbf{g}^{(k)} = (\mathbf{H}_k + a_k\mathbf{z}^{(k)}\mathbf{z}^{(k)\top})\Delta\mathbf{g}^{(k)} = \Delta\mathbf{x}^{(k)}.$$

First note that $\mathbf{z}^{(k)\top}\Delta\mathbf{g}^{(k)}$ is a scalar. Thus,

$$\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)} = (a_k\mathbf{z}^{(k)\top}\Delta\mathbf{g}^{(k)})\mathbf{z}^{(k)},$$

and hence

$$\mathbf{z}^{(k)} = \frac{\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)}}{a_k(\mathbf{z}^{(k)\top}\Delta\mathbf{g}^{(k)})}.$$

We can now determine

$$a_k\mathbf{z}^{(k)}\mathbf{z}^{(k)\top} = \frac{(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})^\top}{a_k(\mathbf{z}^{(k)\top}\Delta\mathbf{g}^{(k)})^2}.$$

Hence,

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})^\top}{a_k(\mathbf{z}^{(k)\top}\Delta\mathbf{g}^{(k)})^2}.$$

The next step is to express the denominator of the second term on the right-hand side of the equation above as a function of the given quantities \mathbf{H}_k , $\Delta\mathbf{g}^{(k)}$, and $\Delta\mathbf{x}^{(k)}$. To accomplish this, premultiply $\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)} = (a_k\mathbf{z}^{(k)\top}\Delta\mathbf{g}^{(k)})\mathbf{z}^{(k)}$ by $\Delta\mathbf{g}^{(k)\top}$ to obtain

$$\Delta\mathbf{g}^{(k)\top}\Delta\mathbf{x}^{(k)} - \Delta\mathbf{g}^{(k)\top}\mathbf{H}_k\Delta\mathbf{g}^{(k)} = \Delta\mathbf{g}^{(k)\top}a_k\mathbf{z}^{(k)}\mathbf{z}^{(k)\top}\Delta\mathbf{g}^{(k)}.$$

Observe that a_k is a scalar and so is $\Delta\mathbf{g}^{(k)\top}\mathbf{z}^{(k)} = \mathbf{z}^{(k)\top}\Delta\mathbf{g}^{(k)}$. Thus,

$$\Delta\mathbf{g}^{(k)\top}\Delta\mathbf{x}^{(k)} - \Delta\mathbf{g}^{(k)\top}\mathbf{H}_k\Delta\mathbf{g}^{(k)} = a_k(\mathbf{z}^{(k)\top}\Delta\mathbf{g}^{(k)})^2.$$

Taking this relation into account yields

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})^\top}{\Delta\mathbf{g}^{(k)\top}(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)})}.$$

We summarize the above development in the following algorithm.

Rank One Algorithm

1. Set $k := 0$; select $\mathbf{x}^{(0)}$ and a real symmetric positive definite \mathbf{H}_0 .
2. If $\mathbf{g}^{(k)} = \mathbf{0}$, stop; else, $\mathbf{d}^{(k)} = -\mathbf{H}_k\mathbf{g}^{(k)}$.

3. Compute

$$\begin{aligned}\alpha_k &= \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}), \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}.\end{aligned}$$

4. Compute

$$\begin{aligned}\Delta \mathbf{x}^{(k)} &= \alpha_k \mathbf{d}^{(k)}, \\ \Delta \mathbf{g}^{(k)} &= \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}, \\ \mathbf{H}_{k+1} &= \mathbf{H}_k + \frac{(\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)}) (\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)})^\top}{\Delta \mathbf{g}^{(k)^\top} (\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)})}.\end{aligned}$$

5. Set $k := k + 1$; go to step 2.

The rank one algorithm is based on satisfying the equation

$$\mathbf{H}_{k+1} \Delta \mathbf{g}^{(k)} = \Delta \mathbf{x}^{(k)}.$$

However, what we want is

$$\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}, \quad i = 0, 1, \dots, k.$$

It turns out that the above is, in fact, true automatically, as stated in the following theorem.

Theorem 11.2 *For the rank one algorithm applied to the quadratic with Hessian $\mathbf{Q} = \mathbf{Q}^\top$, we have $\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}$, $0 \leq i \leq k$.* \square

Proof. We prove the result by induction. From the discussion before the theorem, it is clear that the claim is true for $k = 0$. Suppose now that the theorem is true for $k - 1 \geq 0$; that is, $\mathbf{H}_k \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}$, $i < k$. We now show that the theorem is true for k . Our construction of the correction term ensures that

$$\mathbf{H}_{k+1} \Delta \mathbf{g}^{(k)} = \Delta \mathbf{x}^{(k)}.$$

So we only have to show that

$$\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}, \quad i < k.$$

To this end, fix $i < k$. We have

$$\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \mathbf{H}_k \Delta \mathbf{g}^{(i)} + \frac{(\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)}) (\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)})^\top}{\Delta \mathbf{g}^{(k)^\top} (\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)})} \Delta \mathbf{g}^{(i)}.$$

By the induction hypothesis, $\mathbf{H}_k \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}$. To complete the proof, it is enough to show that the second term on the right-hand side of the equation above is equal to zero. For this to be true it is enough that

$$(\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)})^\top \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(k)^\top} \Delta \mathbf{g}^{(i)} - \Delta \mathbf{g}^{(k)^\top} \mathbf{H}_k \Delta \mathbf{g}^{(i)} = 0.$$

Indeed, since

$$\Delta \mathbf{g}^{(k)\top} \mathbf{H}_k \Delta \mathbf{g}^{(i)} = \Delta \mathbf{g}^{(k)\top} (\mathbf{H}_k \Delta \mathbf{g}^{(i)}) = \Delta \mathbf{g}^{(k)\top} \Delta \mathbf{x}^{(i)}$$

by the induction hypothesis, and because $\Delta \mathbf{g}^{(k)} = \mathbf{Q} \Delta \mathbf{x}^{(k)}$, we have

$$\Delta \mathbf{g}^{(k)\top} \mathbf{H}_k \Delta \mathbf{g}^{(i)} = \Delta \mathbf{g}^{(k)\top} \Delta \mathbf{x}^{(i)} = \Delta \mathbf{x}^{(k)\top} \mathbf{Q} \Delta \mathbf{x}^{(i)} = \Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(i)}.$$

Hence,

$$(\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)})^\top \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(i)} - \Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(i)} = 0,$$

which completes the proof. ■

Example 11.1 Let

$$f(x_1, x_2) = x_1^2 + \frac{1}{2}x_2^2 + 3.$$

Apply the rank one correction algorithm to minimize f . Use $\mathbf{x}^{(0)} = [1, 2]^\top$ and $\mathbf{H}_0 = \mathbf{I}_2$ (2×2 identity matrix).

We can represent f as

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x} + 3.$$

Thus,

$$\mathbf{g}^{(k)} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}^{(k)}.$$

Because $\mathbf{H}_0 = \mathbf{I}_2$,

$$\mathbf{d}^{(0)} = -\mathbf{g}^{(0)} = [-2, -2]^\top.$$

The objective function is quadratic, and hence

$$\begin{aligned} \alpha_0 &= \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(0)} + \alpha \mathbf{d}^{(0)}) = -\frac{\mathbf{g}^{(0)\top} \mathbf{d}^{(0)}}{\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(0)}} \\ &= \frac{[2, 2] \begin{bmatrix} 2 \\ 2 \end{bmatrix}}{[2, 2] \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix}} = \frac{2}{3}, \end{aligned}$$

and thus

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)} = \left[-\frac{1}{3}, \frac{2}{3} \right]^\top.$$

We then compute

$$\begin{aligned}\Delta \mathbf{x}^{(0)} &= \alpha_0 \mathbf{d}^{(0)} = \left[-\frac{4}{3}, -\frac{4}{3} \right]^\top, \\ \mathbf{g}^{(1)} &= \mathbf{Q} \mathbf{x}^{(1)} = \left[-\frac{2}{3}, \frac{2}{3} \right]^\top, \\ \Delta \mathbf{g}^{(0)} &= \mathbf{g}^{(1)} - \mathbf{g}^{(0)} = \left[-\frac{8}{3}, -\frac{4}{3} \right]^\top.\end{aligned}$$

Because

$$\Delta \mathbf{g}^{(0)\top} (\Delta \mathbf{x}^{(0)} - \mathbf{H}_0 \Delta \mathbf{g}^{(0)}) = \left[-\frac{8}{3}, -\frac{4}{3} \right] \left[\frac{4}{3} \atop 0 \right] = -\frac{32}{9},$$

we obtain

$$\mathbf{H}_1 = \mathbf{H}_0 + \frac{(\Delta \mathbf{x}^{(0)} - \mathbf{H}_0 \Delta \mathbf{g}^{(0)}) (\Delta \mathbf{x}^{(0)} - \mathbf{H}_0 \Delta \mathbf{g}^{(0)})^\top}{\Delta \mathbf{g}^{(0)\top} (\Delta \mathbf{x}^{(0)} - \mathbf{H}_0 \Delta \mathbf{g}^{(0)})} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & 1 \end{bmatrix}.$$

Therefore,

$$\mathbf{d}^{(1)} = -\mathbf{H}_1 \mathbf{g}^{(1)} = \left[\frac{1}{3}, -\frac{2}{3} \right]^\top$$

and

$$\alpha_1 = -\frac{\mathbf{g}^{(1)\top} \mathbf{d}^{(1)}}{\mathbf{d}^{(1)\top} \mathbf{Q} \mathbf{d}^{(1)}} = 1.$$

We now compute

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha_1 \mathbf{d}^{(1)} = [0, 0]^\top.$$

Note that $\mathbf{g}^{(2)} = \mathbf{0}$, and therefore $\mathbf{x}^{(2)} = \mathbf{x}^*$. As expected, the algorithm solves the problem in two steps.

Note that the directions $\mathbf{d}^{(0)}$ and $\mathbf{d}^{(1)}$ are \mathbf{Q} -conjugate, in accordance with Theorem 11.1. ■

Unfortunately, the rank one correction algorithm is not very satisfactory, for several reasons. First, the matrix \mathbf{H}_{k+1} that the rank one algorithm generates may not be positive definite (see Example 11.2 below) and thus $\mathbf{d}^{(k+1)}$ may not be a descent direction. This happens even in the quadratic case (see Example 11.10). Furthermore, if

$$\Delta \mathbf{g}^{(k)} (\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)})$$

is close to zero, then there may be numerical problems in evaluating \mathbf{H}_{k+1} .

Example 11.2 Assume that $\mathbf{H}_k > 0$. It turns out that if $\Delta \mathbf{g}^{(k)\top} (\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)}) > 0$, then $\mathbf{H}_{k+1} > 0$ (see Exercise 11.7). However, if

$\Delta\mathbf{g}^{(k)\top}(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)}) < 0$, then \mathbf{H}_{k+1} may not be positive definite. As an example of what might happen if $\Delta\mathbf{g}^{(k)\top}(\Delta\mathbf{x}^{(k)} - \mathbf{H}_k\Delta\mathbf{g}^{(k)}) < 0$, consider applying the rank one algorithm to the function

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

with an initial point

$$\mathbf{x}^{(0)} = [0.59607, 0.59607]^\top$$

and initial matrix

$$\mathbf{H}_0 = \begin{bmatrix} 0.94913 & 0.14318 \\ 0.14318 & 0.59702 \end{bmatrix}.$$

Note that $\mathbf{H}_0 > 0$. We have

$$\Delta\mathbf{g}^{(0)\top}(\Delta\mathbf{x}^{(0)} - \mathbf{H}_0\Delta\mathbf{g}^{(0)}) = -0.03276$$

and

$$\mathbf{H}_1 = \begin{bmatrix} 0.94481 & 0.23324 \\ 0.23324 & -1.2788 \end{bmatrix}.$$

It is easy to check that \mathbf{H}_1 is not positive definite (it is indefinite, with eigenvalues 0.96901 and -1.3030). ■

Fortunately, alternative algorithms have been developed for updating \mathbf{H}_k . In particular, if we use a “rank two” update, then \mathbf{H}_k is guaranteed to be positive definite for all k , provided that the line search is exact. We discuss this in the next section.

11.4 The DFP Algorithm

The rank two update was originally developed by Davidon in 1959 and was subsequently modified by Fletcher and Powell in 1963: hence the name *DFP algorithm*. The DFP algorithm is also known as the *variable metric algorithm*. We summarize the algorithm below.

DFP Algorithm

1. Set $k := 0$; select $\mathbf{x}^{(0)}$ and a real symmetric positive definite \mathbf{H}_0 .
2. If $\mathbf{g}^{(k)} = \mathbf{0}$, stop; else, $\mathbf{d}^{(k)} = -\mathbf{H}_k\mathbf{g}^{(k)}$.
3. Compute

$$\alpha_k = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} + \alpha\mathbf{d}^{(k)}),$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}.$$

4. Compute

$$\begin{aligned}\Delta \mathbf{x}^{(k)} &= \alpha_k \mathbf{d}^{(k)}, \\ \Delta \mathbf{g}^{(k)} &= \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}, \\ \mathbf{H}_{k+1} &= \mathbf{H}_k + \frac{\Delta \mathbf{x}^{(k)} \Delta \mathbf{x}^{(k)\top}}{\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)}} - \frac{[\mathbf{H}_k \Delta \mathbf{g}^{(k)}][\mathbf{H}_k \Delta \mathbf{g}^{(k)}]^\top}{\Delta \mathbf{g}^{(k)\top} \mathbf{H}_k \Delta \mathbf{g}^{(k)}}.\end{aligned}$$

5. Set $k := k + 1$; go to step 2.

We now show that the DFP algorithm is a quasi-Newton method, in the sense that when applied to quadratic problems, we have $\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}$, $0 \leq i \leq k$.

Theorem 11.3 *In the DFP algorithm applied to the quadratic with Hessian $\mathbf{Q} = \mathbf{Q}^\top$, we have $\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}$, $0 \leq i \leq k$. \square*

Proof. We use induction. For $k = 0$, we have

$$\begin{aligned}\mathbf{H}_1 \Delta \mathbf{g}^{(0)} &= \mathbf{H}_0 \Delta \mathbf{g}^{(0)} + \frac{\Delta \mathbf{x}^{(0)} \Delta \mathbf{x}^{(0)\top}}{\Delta \mathbf{x}^{(0)\top} \Delta \mathbf{g}^{(0)}} \Delta \mathbf{g}^{(0)} - \frac{\mathbf{H}_0 \Delta \mathbf{g}^{(0)} \Delta \mathbf{g}^{(0)\top} \mathbf{H}_0}{\Delta \mathbf{g}^{(0)\top} \mathbf{H}_0 \Delta \mathbf{g}^{(0)}} \Delta \mathbf{g}^{(0)} \\ &= \Delta \mathbf{x}^{(0)}.\end{aligned}$$

Assume that the result is true for $k - 1$; that is, $\mathbf{H}_k \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}$, $0 \leq i \leq k - 1$. We now show that $\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}$, $0 \leq i \leq k$. First, consider $i = k$. We have

$$\begin{aligned}\mathbf{H}_{k+1} \Delta \mathbf{g}^{(k)} &= \mathbf{H}_k \Delta \mathbf{g}^{(k)} + \frac{\Delta \mathbf{x}^{(k)} \Delta \mathbf{x}^{(k)\top}}{\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)}} \Delta \mathbf{g}^{(k)} - \frac{\mathbf{H}_k \Delta \mathbf{g}^{(k)} \Delta \mathbf{g}^{(k)\top} \mathbf{H}_k}{\Delta \mathbf{g}^{(k)\top} \mathbf{H}_k \Delta \mathbf{g}^{(k)}} \Delta \mathbf{g}^{(k)} \\ &= \Delta \mathbf{x}^{(k)}.\end{aligned}$$

It remains to consider the case $i < k$. To this end,

$$\begin{aligned}\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} &= \mathbf{H}_k \Delta \mathbf{g}^{(i)} + \frac{\Delta \mathbf{x}^{(k)} \Delta \mathbf{x}^{(k)\top}}{\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)}} \Delta \mathbf{g}^{(i)} - \frac{\mathbf{H}_k \Delta \mathbf{g}^{(k)} \Delta \mathbf{g}^{(k)\top} \mathbf{H}_k}{\Delta \mathbf{g}^{(k)\top} \mathbf{H}_k \Delta \mathbf{g}^{(k)}} \Delta \mathbf{g}^{(i)} \\ &= \Delta \mathbf{x}^{(i)} + \frac{\Delta \mathbf{x}^{(k)}}{\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)}} (\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(i)}) \\ &\quad - \frac{\mathbf{H}_k \Delta \mathbf{g}^{(k)}}{\Delta \mathbf{g}^{(k)\top} \mathbf{H}_k \Delta \mathbf{g}^{(k)}} (\Delta \mathbf{g}^{(k)\top} \Delta \mathbf{x}^{(i)}).\end{aligned}$$

Now,

$$\begin{aligned}\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(i)} &= \Delta \mathbf{x}^{(k)\top} \mathbf{Q} \Delta \mathbf{x}^{(i)} \\ &= \alpha_k \alpha_i \mathbf{d}^{(k)\top} \mathbf{Q} \mathbf{d}^{(i)} \\ &= 0,\end{aligned}$$

by the induction hypothesis and Theorem 11.1. The same arguments yield $\Delta\mathbf{g}^{(k)\top}\Delta\mathbf{x}^{(i)} = 0$. Hence,

$$\mathbf{H}_{k+1}\Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)},$$

which completes the proof. \blacksquare

By Theorems 11.1 and 11.3 we conclude that the DFP algorithm is a conjugate direction algorithm.

Example 11.3 Locate the minimizer of

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \mathbf{x} - \mathbf{x}^\top \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \mathbf{x} \in \mathbb{R}^2.$$

Use the initial point $\mathbf{x}^{(0)} = [0, 0]^\top$ and $\mathbf{H}_0 = \mathbf{I}_2$.

Note that in this case

$$\mathbf{g}^{(k)} = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \mathbf{x}^{(k)} - \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

Hence,

$$\begin{aligned} \mathbf{g}^{(0)} &= [1, -1]^\top, \\ \mathbf{d}^{(0)} &= -\mathbf{H}_0\mathbf{g}^{(0)} = -\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}. \end{aligned}$$

Because f is a quadratic function,

$$\begin{aligned} \alpha_0 &= \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(0)} + \alpha\mathbf{d}^{(0)}) = -\frac{\mathbf{g}^{(0)\top}\mathbf{d}^{(0)}}{\mathbf{d}^{(0)\top}\mathbf{Q}\mathbf{d}^{(0)}} \\ &= -\frac{[1, -1] \begin{bmatrix} -1 \\ 1 \end{bmatrix}}{[-1, 1] \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix}} = 1. \end{aligned}$$

Therefore,

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0\mathbf{d}^{(0)} = [-1, 1]^\top.$$

We then compute

$$\begin{aligned} \Delta\mathbf{x}^{(0)} &= \mathbf{x}^{(1)} - \mathbf{x}^{(0)} = [-1, 1]^\top, \\ \mathbf{g}^{(1)} &= \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} - \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \end{aligned}$$

and

$$\Delta \mathbf{g}^{(0)} = \mathbf{g}^{(1)} - \mathbf{g}^{(0)} = [-2, 0]^\top.$$

Observe that

$$\begin{aligned}\Delta \mathbf{x}^{(0)} \Delta \mathbf{x}^{(0)\top} &= \begin{bmatrix} -1 \\ 1 \end{bmatrix} [-1, 1] = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \\ \Delta \mathbf{x}^{(0)\top} \Delta \mathbf{g}^{(0)} &= [-1, 1] \begin{bmatrix} -2 \\ 0 \end{bmatrix} = 2, \\ \mathbf{H}_0 \Delta \mathbf{g}^{(0)} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -2 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \end{bmatrix}.\end{aligned}$$

Thus,

$$(\mathbf{H}_0 \Delta \mathbf{g}^{(0)}) (\mathbf{H}_0 \Delta \mathbf{g}^{(0)})^\top = \begin{bmatrix} -2 \\ 0 \end{bmatrix} [-2, 0] = \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix}$$

and

$$\Delta \mathbf{g}^{(0)\top} \mathbf{H}_0 \Delta \mathbf{g}^{(0)} = [-2, 0] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -2 \\ 0 \end{bmatrix} = 4.$$

Using the above, we now compute \mathbf{H}_1 :

$$\begin{aligned}\mathbf{H}_1 &= \mathbf{H}_0 + \frac{\Delta \mathbf{x}^{(0)} \Delta \mathbf{x}^{(0)\top}}{\Delta \mathbf{x}^{(0)\top} \Delta \mathbf{g}^{(0)}} - \frac{(\mathbf{H}_0 \Delta \mathbf{g}^{(0)}) (\mathbf{H}_0 \Delta \mathbf{g}^{(0)})^\top}{\Delta \mathbf{g}^{(0)\top} \mathbf{H}_0 \Delta \mathbf{g}^{(0)}} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} 4 & 0 \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} \end{bmatrix}.\end{aligned}$$

We now compute $\mathbf{d}^{(1)} = -\mathbf{H}_1 \mathbf{g}^{(1)} = [0, 1]^\top$ and

$$\alpha_1 = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(1)} + \alpha \mathbf{d}^{(1)}) = -\frac{\mathbf{g}^{(1)\top} \mathbf{d}^{(1)}}{\mathbf{d}^{(1)\top} \mathbf{Q} \mathbf{d}^{(1)}} = \frac{1}{2}.$$

Hence,

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha_1 \mathbf{d}^{(1)} = [-1, 3/2]^\top = \mathbf{x}^*,$$

because f is a quadratic function of two variables.

Note that we have $\mathbf{d}^{(0)\top} \mathbf{Q} \mathbf{d}^{(1)} = \mathbf{d}^{(1)\top} \mathbf{Q} \mathbf{d}^{(0)} = 0$; that is, $\mathbf{d}^{(0)}$ and $\mathbf{d}^{(1)}$ are \mathbf{Q} -conjugate directions. ■

We now show that in the DFP algorithm, \mathbf{H}_{k+1} inherits positive definiteness from \mathbf{H}_k .

Theorem 11.4 Suppose that $\mathbf{g}^{(k)} \neq \mathbf{0}$. In the DFP algorithm, if \mathbf{H}_k is positive definite, then so is \mathbf{H}_{k+1} . \square

Proof. We first write the following quadratic form:

$$\begin{aligned}\mathbf{x}^\top \mathbf{H}_{k+1} \mathbf{x} &= \mathbf{x}^\top \mathbf{H}_k \mathbf{x} + \frac{\mathbf{x}^\top \Delta \mathbf{x}^{(k)} \Delta \mathbf{x}^{(k)\top} \mathbf{x}}{\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)}} - \frac{\mathbf{x}^\top (\mathbf{H}_k \Delta \mathbf{g}^{(k)}) (\mathbf{H}_k \Delta \mathbf{g}^{(k)})^\top \mathbf{x}}{\Delta \mathbf{g}^{(k)\top} \mathbf{H}_k \Delta \mathbf{g}^{(k)}} \\ &= \mathbf{x}^\top \mathbf{H}_k \mathbf{x} + \frac{(\mathbf{x}^\top \Delta \mathbf{x}^{(k)})^2}{\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)}} - \frac{(\mathbf{x}^\top \mathbf{H}_k \Delta \mathbf{g}^{(k)})^2}{\Delta \mathbf{g}^{(k)\top} \mathbf{H}_k \Delta \mathbf{g}^{(k)}}.\end{aligned}$$

Define

$$\begin{aligned}\mathbf{a} &\triangleq \mathbf{H}_k^{1/2} \mathbf{x}, \\ \mathbf{b} &\triangleq \mathbf{H}_k^{1/2} \Delta \mathbf{g}^{(k)},\end{aligned}$$

where

$$\mathbf{H}_k = \mathbf{H}_k^{1/2} \mathbf{H}_k^{1/2}.$$

Note that because $\mathbf{H}_k > 0$, its square root is well-defined; see Section 3.4 for more information on this property of positive definite matrices. Using the definitions of \mathbf{a} and \mathbf{b} , we obtain

$$\begin{aligned}\mathbf{x}^\top \mathbf{H}_k \mathbf{x} &= \mathbf{x}^\top \mathbf{H}_k^{1/2} \mathbf{H}_k^{1/2} \mathbf{x} = \mathbf{a}^\top \mathbf{a}, \\ \mathbf{x}^\top \mathbf{H}_k \Delta \mathbf{g}^{(k)} &= \mathbf{x}^\top \mathbf{H}_k^{1/2} \mathbf{H}_k^{1/2} \Delta \mathbf{g}^{(k)} = \mathbf{a}^\top \mathbf{b},\end{aligned}$$

and

$$\Delta \mathbf{g}^{(k)\top} \mathbf{H}_k \Delta \mathbf{g}^{(k)} = \Delta \mathbf{g}^{(k)\top} \mathbf{H}_k^{1/2} \mathbf{H}_k^{1/2} \Delta \mathbf{g}^{(k)} = \mathbf{b}^\top \mathbf{b}.$$

Hence,

$$\begin{aligned}\mathbf{x}^\top \mathbf{H}_{k+1} \mathbf{x} &= \mathbf{a}^\top \mathbf{a} + \frac{(\mathbf{x}^\top \Delta \mathbf{x}^{(k)})^2}{\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)}} - \frac{(\mathbf{a}^\top \mathbf{b})^2}{\mathbf{b}^\top \mathbf{b}} \\ &= \frac{\|\mathbf{a}\|^2 \|\mathbf{b}\|^2 - (\langle \mathbf{a}, \mathbf{b} \rangle)^2}{\|\mathbf{b}\|^2} + \frac{(\mathbf{x}^\top \Delta \mathbf{x}^{(k)})^2}{\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)}}.\end{aligned}$$

We also have

$$\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)} = \Delta \mathbf{x}^{(k)\top} (\mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}) = -\Delta \mathbf{x}^{(k)\top} \mathbf{g}^{(k)},$$

since $\Delta \mathbf{x}^{(k)\top} \mathbf{g}^{(k+1)} = \alpha_k \mathbf{d}^{(k)\top} \mathbf{g}^{(k+1)} = 0$ by Lemma 10.2 (see also Exercise 11.1). Because

$$\Delta \mathbf{x}^{(k)} = \alpha_k \mathbf{d}^{(k)} = -\alpha_k \mathbf{H}_k \mathbf{g}^{(k)},$$

we have

$$\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)} = -\Delta \mathbf{x}^{(k)\top} \mathbf{g}^{(k)} = \alpha_k \mathbf{g}^{(k)\top} \mathbf{H}_k \mathbf{g}^{(k)}.$$

This yields

$$\mathbf{x}^\top \mathbf{H}_{k+1} \mathbf{x} = \frac{\|\mathbf{a}\|^2 \|\mathbf{b}\|^2 - (\langle \mathbf{a}, \mathbf{b} \rangle)^2}{\|\mathbf{b}\|^2} + \frac{(\mathbf{x}^\top \Delta \mathbf{x}^{(k)})^2}{\alpha_k \mathbf{g}^{(k)\top} \mathbf{H}_k \mathbf{g}^{(k)}}.$$

Both terms on the right-hand side of the above equation are nonnegative—the first term is nonnegative because of the Cauchy-Schwarz inequality, and the second term is nonnegative because $\mathbf{H}_k > 0$ and $\alpha_k > 0$ (by Proposition 11.1). Therefore, to show that $\mathbf{x}^\top \mathbf{H}_{k+1} \mathbf{x} > 0$ for $\mathbf{x} \neq \mathbf{0}$, we only need to demonstrate that these terms do not both vanish simultaneously.

The first term vanishes only if \mathbf{a} and \mathbf{b} are proportional, that is, if $\mathbf{a} = \beta \mathbf{b}$ for some scalar β . Thus, to complete the proof it is enough to show that if $\mathbf{a} = \beta \mathbf{b}$, then $(\mathbf{x}^\top \Delta \mathbf{x}^{(k)})^2 / (\alpha_k \mathbf{g}^{(k)\top} \mathbf{H}_k \mathbf{g}^{(k)}) > 0$. Indeed, first observe that

$$\mathbf{H}_k^{1/2} \mathbf{x} = \mathbf{a} = \beta \mathbf{b} = \beta \mathbf{H}_k^{1/2} \Delta \mathbf{g}^{(k)} = \mathbf{H}_k^{1/2} (\beta \Delta \mathbf{g}^{(k)}).$$

Hence,

$$\mathbf{x} = \beta \Delta \mathbf{g}^{(k)}.$$

Using the expression for \mathbf{x} above and the expression $\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)} = \alpha_k \mathbf{g}^{(k)\top} \mathbf{H}_k \mathbf{g}^{(k)}$, we obtain

$$\begin{aligned} \frac{(\mathbf{x}^\top \Delta \mathbf{x}^{(k)})^2}{\alpha_k \mathbf{g}^{(k)\top} \mathbf{H}_k \mathbf{g}^{(k)}} &= \frac{\beta^2 (\Delta \mathbf{g}^{(k)\top} \Delta \mathbf{x}^{(k)})^2}{\alpha_k \mathbf{g}^{(k)\top} \mathbf{H}_k \mathbf{g}^{(k)}} = \frac{\beta^2 (\alpha_k \mathbf{g}^{(k)\top} \mathbf{H}_k \mathbf{g}^{(k)})^2}{\alpha_k \mathbf{g}^{(k)\top} \mathbf{H}_k \mathbf{g}^{(k)}} \\ &= \beta^2 \alpha_k \mathbf{g}^{(k)\top} \mathbf{H}_k \mathbf{g}^{(k)} > 0. \end{aligned}$$

Thus, for all $\mathbf{x} \neq \mathbf{0}$,

$$\mathbf{x}^\top \mathbf{H}_{k+1} \mathbf{x} > 0,$$

which completes the proof. ■

The DFP algorithm is superior to the rank one algorithm in that it preserves the positive definiteness of \mathbf{H}_k . However, it turns out that in the case of larger nonquadratic problems the algorithm has the tendency of sometimes getting “stuck.” This phenomenon is attributed to \mathbf{H}_k becoming nearly singular [19]. In the next section we discuss an algorithm that alleviates this problem.

11.5 The BFGS Algorithm

In 1970, an alternative update formula was suggested independently by Broyden, Fletcher, Goldfarb, and Shanno. The method, now called the *BFGS algorithm*, is discussed in this section.

To derive the BFGS update, we use the concept of *duality*, or *complementarity*, as presented in [43] and [88]. To discuss this concept, recall that the

updating formulas for the approximation of the inverse of the Hessian matrix were based on satisfying the equations

$$\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}, \quad 0 \leq i \leq k,$$

which were derived from $\Delta \mathbf{g}^{(i)} = \mathbf{Q} \Delta \mathbf{x}^{(i)}$, $0 \leq i \leq k$. We then formulated update formulas for the approximations to the inverse of the Hessian matrix \mathbf{Q}^{-1} . An alternative to approximating \mathbf{Q}^{-1} is to approximate \mathbf{Q} itself. To do this let \mathbf{B}_k be our estimate of \mathbf{Q} at the k th step. We require \mathbf{B}_{k+1} to satisfy

$$\Delta \mathbf{g}^{(i)} = \mathbf{B}_{k+1} \Delta \mathbf{x}^{(i)}, \quad 0 \leq i \leq k.$$

Notice that this set of equations is similar to the previous set of equations for \mathbf{H}_{k+1} , the only difference being that the roles of $\Delta \mathbf{x}^{(i)}$ and $\Delta \mathbf{g}^{(i)}$ are interchanged. Thus, given any update formula for \mathbf{H}_k , a corresponding update formula for \mathbf{B}_k can be found by interchanging the roles of \mathbf{B}_k and \mathbf{H}_k and of $\Delta \mathbf{g}^{(k)}$ and $\Delta \mathbf{x}^{(k)}$. In particular, the BFGS update for \mathbf{B}_k corresponds to the DFP update for \mathbf{H}_k . Formulas related in this way are said to be *dual* or *complementary* [43].

Recall that the DFP update for the approximation \mathbf{H}_k of the inverse Hessian is

$$\mathbf{H}_{k+1}^{DFP} = \mathbf{H}_k + \frac{\Delta \mathbf{x}^{(k)} \Delta \mathbf{x}^{(k)\top}}{\Delta \mathbf{x}^{(k)\top} \Delta \mathbf{g}^{(k)}} - \frac{\mathbf{H}_k \Delta \mathbf{g}^{(k)} \Delta \mathbf{g}^{(k)\top} \mathbf{H}_k}{\Delta \mathbf{g}^{(k)\top} \mathbf{H}_k \Delta \mathbf{g}^{(k)}}.$$

Using the complementarity concept, we can easily obtain an update equation for the approximation \mathbf{B}_k of the Hessian:

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{\Delta \mathbf{g}^{(k)} \Delta \mathbf{g}^{(k)\top}}{\Delta \mathbf{g}^{(k)\top} \Delta \mathbf{x}^{(k)}} - \frac{\mathbf{B}_k \Delta \mathbf{x}^{(k)} \Delta \mathbf{x}^{(k)\top} \mathbf{B}_k}{\Delta \mathbf{x}^{(k)\top} \mathbf{B}_k \Delta \mathbf{x}^{(k)}}.$$

This is the BFGS update of \mathbf{B}_k .

Now, to obtain the BFGS update for the approximation of the inverse Hessian, we take the inverse of \mathbf{B}_{k+1} to obtain

$$\begin{aligned} \mathbf{H}_{k+1}^{BFGS} &= (\mathbf{B}_{k+1})^{-1} \\ &= \left(\mathbf{B}_k + \frac{\Delta \mathbf{g}^{(k)} \Delta \mathbf{g}^{(k)\top}}{\Delta \mathbf{g}^{(k)\top} \Delta \mathbf{x}^{(k)}} - \frac{\mathbf{B}_k \Delta \mathbf{x}^{(k)} \Delta \mathbf{x}^{(k)\top} \mathbf{B}_k}{\Delta \mathbf{x}^{(k)\top} \mathbf{B}_k \Delta \mathbf{x}^{(k)}} \right)^{-1}. \end{aligned}$$

To compute \mathbf{H}_{k+1}^{BFGS} by inverting the right-hand side of this equation, we apply the following formula for a matrix inverse, known as the *Sherman-Morrison formula* (see [63, p. 123] or [53, p. 50]).

Lemma 11.1 *Let \mathbf{A} be a nonsingular matrix. Let \mathbf{u} and \mathbf{v} be column vectors such that $1 + \mathbf{v}^\top \mathbf{A}^{-1} \mathbf{u} \neq 0$. Then, $\mathbf{A} + \mathbf{u}\mathbf{v}^\top$ is nonsingular, and its inverse can be written in terms of \mathbf{A}^{-1} using the following formula:*

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^\top)^{-1} = \mathbf{A}^{-1} - \frac{(\mathbf{A}^{-1}\mathbf{u})(\mathbf{v}^\top \mathbf{A}^{-1})}{1 + \mathbf{v}^\top \mathbf{A}^{-1}\mathbf{u}}.$$

□

Proof. We can prove the result easily by verification. ■

From Lemma 11.1 it follows that if \mathbf{A}^{-1} is known, then the inverse of the matrix \mathbf{A} augmented by a rank one matrix can be obtained by a modification of the matrix \mathbf{A}^{-1} .

Applying Lemma 11.1 twice to \mathbf{B}_{k+1} (see Exercise 11.12) yields

$$\begin{aligned} \mathbf{H}_{k+1}^{BFGS} &= \mathbf{H}_k + \left(1 + \frac{\Delta\mathbf{g}^{(k)\top} \mathbf{H}_k \Delta\mathbf{g}^{(k)}}{\Delta\mathbf{g}^{(k)\top} \Delta\mathbf{x}^{(k)}}\right) \frac{\Delta\mathbf{x}^{(k)} \Delta\mathbf{x}^{(k)\top}}{\Delta\mathbf{x}^{(k)\top} \Delta\mathbf{g}^{(k)}} \\ &\quad - \frac{\mathbf{H}_k \Delta\mathbf{g}^{(k)} \Delta\mathbf{x}^{(k)\top} + (\mathbf{H}_k \Delta\mathbf{g}^{(k)} \Delta\mathbf{x}^{(k)\top})^\top}{\Delta\mathbf{g}^{(k)\top} \Delta\mathbf{x}^{(k)}}, \end{aligned}$$

which represents the BFGS formula for updating \mathbf{H}_k .

Recall that for the quadratic case the DFP algorithm satisfies $\mathbf{H}_{k+1}^{DFP} \Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}$, $0 \leq i \leq k$. Therefore, the BFGS update for \mathbf{B}_k satisfies $\mathbf{B}_{k+1} \Delta\mathbf{x}^{(i)} = \Delta\mathbf{g}^{(i)}$, $0 \leq i \leq k$. By construction of the BFGS formula for \mathbf{H}_{k+1}^{BFGS} , we conclude that $\mathbf{H}_{k+1}^{BFGS} \Delta\mathbf{g}^{(i)} = \Delta\mathbf{x}^{(i)}$, $0 \leq i \leq k$. Hence, the BFGS algorithm enjoys all the properties of quasi-Newton methods, including the conjugate directions property. Moreover, the BFGS algorithm also inherits the positive definiteness property of the DFP algorithm; that is, if $\mathbf{g}^{(k)} \neq \mathbf{0}$ and $\mathbf{H}_k > 0$, then $\mathbf{H}_{k+1}^{BFGS} > 0$.

The BFGS update is reasonably robust when the line searches are sloppy (see [19]). This property allows us to save time in the line search part of the algorithm. The BFGS formula is often far more efficient than the DFP formula (see [107] for further discussion).

We conclude our discussion of the BFGS algorithm with the following numerical example.

Example 11.4 Use the BFGS method to minimize

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} - \mathbf{x}^\top \mathbf{b} + \log(\pi),$$

where

$$\mathbf{Q} = \begin{bmatrix} 5 & -3 \\ -3 & 2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Take $\mathbf{H}_0 = \mathbf{I}_2$ and $\mathbf{x}^{(0)} = [0, 0]^\top$. Verify that $\mathbf{H}_2 = \mathbf{Q}^{-1}$.

We have

$$\mathbf{d}^{(0)} = -\mathbf{g}^{(0)} = -(\mathbf{Q}\mathbf{x}^{(0)} - \mathbf{b}) = \mathbf{b} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

The objective function is a quadratic, and hence we can use the following formula to compute α_0 :

$$\alpha_0 = -\frac{\mathbf{g}^{(0)\top} \mathbf{d}^{(0)}}{\mathbf{d}^{(0)\top} Q \mathbf{d}^{(0)}} = \frac{1}{2}.$$

Therefore,

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{d}^{(0)} = \begin{bmatrix} 0 \\ 1/2 \end{bmatrix}.$$

To compute $\mathbf{H}_1 = \mathbf{H}_1^{BFGS}$, we need the following quantities:

$$\begin{aligned} \Delta \mathbf{x}^{(0)} &= \mathbf{x}^{(1)} - \mathbf{x}^{(0)} = \begin{bmatrix} 0 \\ 1/2 \end{bmatrix}, \\ \mathbf{g}^{(1)} &= Q \mathbf{x}^{(1)} - \mathbf{b} = \begin{bmatrix} -3/2 \\ 0 \end{bmatrix}, \\ \Delta \mathbf{g}^{(0)} &= \mathbf{g}^{(1)} - \mathbf{g}^{(0)} = \begin{bmatrix} -3/2 \\ 1 \end{bmatrix}. \end{aligned}$$

Therefore,

$$\begin{aligned} \mathbf{H}_1 &= \mathbf{H}_0 + \left(1 + \frac{\Delta \mathbf{g}^{(0)\top} \mathbf{H}_0 \Delta \mathbf{g}^{(0)}}{\Delta \mathbf{g}^{(0)\top} \Delta \mathbf{x}^{(0)}} \right) \frac{\Delta \mathbf{x}^{(0)} \Delta \mathbf{x}^{(0)\top}}{\Delta \mathbf{x}^{(0)\top} \Delta \mathbf{g}^{(0)}} \\ &\quad - \frac{\Delta \mathbf{x}^{(0)} \Delta \mathbf{g}^{(0)\top} \mathbf{H}_0 + \mathbf{H}_0 \Delta \mathbf{g}^{(0)} \Delta \mathbf{x}^{(0)\top}}{\Delta \mathbf{g}^{(0)\top} \Delta \mathbf{x}^{(0)}} \\ &= \begin{bmatrix} 1 & 3/2 \\ 3/2 & 11/4 \end{bmatrix}. \end{aligned}$$

Hence, we have

$$\begin{aligned} \mathbf{d}^{(1)} &= -\mathbf{H}_1 \mathbf{g}^{(1)} = \begin{bmatrix} 3/2 \\ 9/4 \end{bmatrix}, \\ \alpha_1 &= -\frac{\mathbf{g}^{(1)\top} \mathbf{d}^{(1)}}{\mathbf{d}^{(1)\top} Q \mathbf{d}^{(1)}} = 2. \end{aligned}$$

Therefore,

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \alpha_1 \mathbf{d}^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}.$$

Because our objective function is a quadratic on \mathbb{R}^2 , $\mathbf{x}^{(2)}$ is the minimizer. Notice that the gradient at $\mathbf{x}^{(2)}$ is $\mathbf{0}$; that is, $\mathbf{g}^{(2)} = \mathbf{0}$.

To verify that $\mathbf{H}_2 = \mathbf{Q}^{-1}$, we compute

$$\begin{aligned}\Delta\mathbf{x}^{(1)} &= \mathbf{x}^{(2)} - \mathbf{x}^{(1)} = \begin{bmatrix} 3 \\ 9/2 \end{bmatrix}, \\ \Delta\mathbf{g}^{(1)} &= \mathbf{g}^{(2)} - \mathbf{g}^{(1)} = \begin{bmatrix} 3/2 \\ 0 \end{bmatrix}.\end{aligned}$$

Hence,

$$\begin{aligned}\mathbf{H}_2 &= \mathbf{H}_1 + \left(1 + \frac{\Delta\mathbf{g}^{(1)\top} \mathbf{H}_1 \Delta\mathbf{g}^{(1)}}{\Delta\mathbf{g}^{(1)\top} \Delta\mathbf{x}^{(1)}}\right) \frac{\Delta\mathbf{x}^{(1)} \Delta\mathbf{x}^{(1)\top}}{\Delta\mathbf{x}^{(1)\top} \Delta\mathbf{g}^{(1)}} \\ &\quad - \frac{\Delta\mathbf{x}^{(1)} \Delta\mathbf{g}^{(1)\top} \mathbf{H}_1 + \mathbf{H}_1 \Delta\mathbf{g}^{(1)} \Delta\mathbf{x}^{(1)\top}}{\Delta\mathbf{g}^{(1)\top} \Delta\mathbf{x}^{(1)}} \\ &= \begin{bmatrix} 2 & 3 \\ 3 & 5 \end{bmatrix}.\end{aligned}$$

Note that indeed $\mathbf{H}_2 \mathbf{Q} = \mathbf{Q} \mathbf{H}_2 = \mathbf{I}_2$, and hence $\mathbf{H}_2 = \mathbf{Q}^{-1}$. ■

For nonquadratic problems, quasi-Newton algorithms will not usually converge in n steps. As in the case of the conjugate gradient methods, here, too, some modifications may be necessary to deal with nonquadratic problems. For example, we may reinitialize the direction vector to the negative gradient after every few iterations (e.g., n or $n+1$), and continue until the algorithm satisfies the stopping criterion.

EXERCISES

11.1 Given $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $f \in \mathcal{C}^1$, consider the algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)},$$

where $\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, \dots$ are vectors in \mathbb{R}^n , and $\alpha_k \geq 0$ is chosen to minimize $f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)})$; that is,

$$\alpha_k = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}).$$

Note that the general algorithm above encompasses almost all algorithms that we discussed in this part, including the steepest descent, Newton, conjugate gradient, and quasi-Newton algorithms.

Let $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$, and assume that $\mathbf{d}^{(k)\top} \mathbf{g}^{(k)} < 0$.

- a. Show that $\mathbf{d}^{(k)}$ is a descent direction for f in the sense that there exists $\bar{\alpha} > 0$ such that for all $\alpha \in (0, \bar{\alpha}]$,

$$f(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)}).$$

- b. Show that $\alpha_k > 0$.
- c. Show that $\mathbf{d}^{(k)\top} \mathbf{g}^{(k+1)} = 0$.
- d. Show that the following algorithms all satisfy the condition $\mathbf{d}^{(k)\top} \mathbf{g}^{(k)} < 0$, if $\mathbf{g}^{(k)} \neq \mathbf{0}$:
1. Steepest descent algorithm.
 2. Newton's method, assuming that the Hessian is positive definite.
 3. Conjugate gradient algorithm.
 4. Quasi-Newton algorithm, assuming that $\mathbf{H}_k > 0$.
- e. For the case where $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b}$, with $\mathbf{Q} = \mathbf{Q}^\top > 0$, derive an expression for α_k in terms of \mathbf{Q} , $\mathbf{d}^{(k)}$, and $\mathbf{g}^{(k)}$.

11.2 Consider Newton's algorithm applied to a function $f \in \mathcal{C}^2$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{F}(\mathbf{x}^{(k)})^{-1} \nabla f(\mathbf{x}^{(k)}),$$

where α_k is chosen according to a line search. Is this algorithm a member of the quasi-Newton family?

11.3 In some optimization methods, when minimizing a given function $f(\mathbf{x})$, we select an initial guess $\mathbf{x}^{(0)}$ and a real symmetric positive definite matrix \mathbf{H}_0 . Then we iteratively compute \mathbf{H}_k , $\mathbf{d}^{(k)} = -\mathbf{H}_k \mathbf{g}^{(k)}$ (where $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$), and $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$, where

$$\alpha_k = \arg \min_{\alpha \geq 0} f\left(\mathbf{x}^{(k)} + \alpha \mathbf{d}^{(k)}\right).$$

Suppose that the function we wish to minimize is a standard quadratic of the form

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b} + c, \quad \mathbf{Q} = \mathbf{Q}^\top > 0.$$

- a. Find an expression for α_k in terms of \mathbf{Q} , \mathbf{H}_k , $\mathbf{g}^{(k)}$, and $\mathbf{d}^{(k)}$;
- b. Give a sufficient condition on \mathbf{H}_k for α_k to be positive.

11.4 Consider the algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{H} \mathbf{g}^{(k)},$$

where, as usual, $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$ and \mathbf{H} is a fixed symmetric matrix.

- a. Suppose that $f \in \mathcal{C}^3$ and there is a point \mathbf{x}^* such that $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\mathbf{F}(\mathbf{x}^*)^{-1}$ exists. Find \mathbf{H} such that if $\mathbf{x}^{(0)}$ is sufficiently close to \mathbf{x}^* , then $\mathbf{x}^{(k)}$ converges to \mathbf{x}^* with order of convergence of at least 2.
- b. With the setting of \mathbf{H} in part a, is the given algorithm a quasi-Newton method?

11.5 Minimize the function

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \mathbf{x} - \mathbf{x}^\top \begin{bmatrix} 1 \\ -1 \end{bmatrix} + 7$$

using the rank one correction method with the starting point $\mathbf{x}^{(0)} = \mathbf{0}$.

11.6 Consider the algorithm

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{M}_k \nabla f(\mathbf{x}^{(k)}),$$

where $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, $f \in \mathcal{C}^1$, $\mathbf{M}_k \in \mathbb{R}^{2 \times 2}$ is given by

$$\mathbf{M}_k = \begin{bmatrix} 1 & 0 \\ 0 & a \end{bmatrix}$$

with $a \in \mathbb{R}$, and

$$\alpha_k = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} - \alpha \mathbf{M}_k \nabla f(\mathbf{x}^{(k)})).$$

Suppose that at some iteration k we have $\nabla f(\mathbf{x}^{(k)}) = [1, 1]^\top$. Find the largest range of values of a that guarantees that $\alpha_k > 0$ for any f .

11.7 Consider the rank one algorithm. Assume that $\mathbf{H}_k > 0$. Show that if $\Delta \mathbf{g}^{(k)\top} (\Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)}) > 0$, then $\mathbf{H}_{k+1} > 0$.

11.8 Based on the rank one update equation, derive an update formula using complementarity and the matrix inverse formula.

11.9 Let

$$\begin{aligned} f &= \frac{1}{2} \mathbf{x}^\top \mathbf{Q} \mathbf{x} - \mathbf{x}^\top \mathbf{b} + c \\ &= \frac{1}{2} \mathbf{x}^\top \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \mathbf{x} - \mathbf{x}^\top \begin{bmatrix} 1 \\ -1 \end{bmatrix} + 7 \end{aligned}$$

and $\mathbf{x}^{(0)} = \mathbf{0}$. Use the rank one correction method to generate two \mathbf{Q} -conjugate directions.

11.10 Apply the rank one algorithm to the problem in Example 11.3.

11.11 Consider the DFP algorithm applied to the quadratic function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} - \mathbf{x}^\top \mathbf{b},$$

where $\mathbf{Q} = \mathbf{Q}^\top > 0$.

- a. Write down a formula for α_k in terms of \mathbf{Q} , $\mathbf{g}^{(k)}$, and $\mathbf{d}^{(k)}$.
- b. Show that if $\mathbf{g}^{(k)} \neq \mathbf{0}$, then $\alpha_k > 0$.

11.12 Use Lemma 11.1 to derive the BFGS update formula based on the DFP formula, using complementarity.

Hint: Define

$$\begin{aligned} \mathbf{A}_0 &= \mathbf{B}_k, \\ \mathbf{u}_0 &= \frac{\Delta\mathbf{g}^{(k)}}{\Delta\mathbf{g}^{(k)\top}\Delta\mathbf{x}^{(k)}}, \\ \mathbf{v}_0^\top &= \Delta\mathbf{g}^{(k)\top}, \\ \mathbf{u}_1 &= -\frac{\mathbf{B}_k\Delta\mathbf{x}^{(k)}}{\Delta\mathbf{x}^{(k)\top}\mathbf{B}_k\Delta\mathbf{x}^{(k)}}, \\ \mathbf{v}_1^\top &= \Delta\mathbf{x}^{(k)\top}\mathbf{B}_k, \\ \mathbf{A}_1 &= \mathbf{B}_k + \frac{\Delta\mathbf{g}^{(k)}\Delta\mathbf{g}^{(k)\top}}{\Delta\mathbf{g}^{(k)\top}\Delta\mathbf{x}^{(k)}} = \mathbf{A}_0 + \mathbf{u}_0\mathbf{v}_0^\top. \end{aligned}$$

Using the notation above, represent \mathbf{B}_{k+1} as

$$\begin{aligned} \mathbf{B}_{k+1} &= \mathbf{A}_0 + \mathbf{u}_0\mathbf{v}_0^\top + \mathbf{u}_1\mathbf{v}_1^\top \\ &= \mathbf{A}_1 + \mathbf{u}_1\mathbf{v}_1^\top. \end{aligned}$$

Apply Lemma 11.1 to the above.

11.13 Assuming exact line search, show that if $\mathbf{H}_0 = \mathbf{I}_n$ ($n \times n$ identity matrix), then the first two steps of the BFGS algorithm yield the same points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ as conjugate gradient algorithms with the Hestenes-Stiefel, the Polak-Ribi  re, and the Fletcher-Reeves formulas.

11.14 Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be such that $f \in \mathcal{C}^1$. Consider an optimization algorithm applied to this f , of the usual form $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$, where

$\alpha_k \geq 0$ is chosen according to line search. Suppose that $\mathbf{d}^{(k)} = -\mathbf{H}_k \mathbf{g}^{(k)}$, where $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$ and \mathbf{H}_k is symmetric.

- a. Show that if \mathbf{H}_k satisfies the following conditions whenever the algorithm is applied to a quadratic, then the algorithm is quasi-Newton:
 1. $\mathbf{H}_{k+1} = \mathbf{H}_k + \mathbf{U}_k$.
 2. $\mathbf{U}_k \Delta \mathbf{g}^{(k)} = \Delta \mathbf{x}^{(k)} - \mathbf{H}_k \Delta \mathbf{g}^{(k)}$.
 3. $\mathbf{U}_k = \mathbf{a}^{(k)} \Delta \mathbf{x}^{(k)\top} + \mathbf{b}^{(k)} \Delta \mathbf{g}^{(k)\top} \mathbf{H}_k$, where $\mathbf{a}^{(k)}$ and $\mathbf{b}^{(k)}$ are in \mathbb{R}^n .
- b. Which (if any) among the rank-one, DFP, and BFGS algorithms satisfy the three conditions in part a (whenever the algorithm is applied to a quadratic)? For those that do, specify the vectors $\mathbf{a}^{(k)}$ and $\mathbf{b}^{(k)}$.

11.15 Given a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$, consider an algorithm $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \mathbf{H}_k \mathbf{g}^{(k)}$ for finding the minimizer of f , where $\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k)})$ and $\mathbf{H}_k \in \mathbb{R}^{n \times n}$ is symmetric. Suppose that $\mathbf{H}_k = \phi \mathbf{H}_k^{DFP} + (1 - \phi) \mathbf{H}_k^{BFGS}$, where $\phi \in \mathbb{R}$, and \mathbf{H}_k^{DFP} and \mathbf{H}_k^{BFGS} are matrices generated by the DFP and BFGS algorithms, respectively.

- a. Show that the algorithm above is a quasi-Newton algorithm. Is the above algorithm a conjugate direction algorithm?
- b. Suppose that $0 \leq \phi \leq 1$. Show that if $\mathbf{H}_0^{DFP} > 0$ and $\mathbf{H}_0^{BFGS} > 0$, then $\mathbf{H}_k > 0$ for all k . What can you conclude from this about whether or not the algorithm has the descent property?

11.16 Consider the following simple modification of the quasi-Newton family of algorithms. In the quadratic case, instead of the usual quasi-Newton condition $\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \Delta \mathbf{x}^{(i)}$, $0 \leq i \leq k$, suppose that we have $\mathbf{H}_{k+1} \Delta \mathbf{g}^{(i)} = \rho_i \Delta \mathbf{x}^{(i)}$, $0 \leq i \leq k$, where $\rho_i > 0$. We refer to the set of algorithms that satisfy the condition above as the *symmetric Huang family*.

Show that the symmetric Huang family algorithms are conjugate direction algorithms.

11.17 Write a MATLAB program to implement the quasi-Newton algorithm for general functions. Use the secant method for the line search (e.g., the MATLAB function of Exercise 7.11). Test the various update formulas for \mathbf{H}_k on Rosenbrock's function (see Exercise 9.4), with an initial condition $\mathbf{x}^{(0)} = [-2, 2]^\top$. For this exercise, reinitialize the update direction to the negative gradient every six iterations.

11.18 Consider the function

$$f(\mathbf{x}) = \frac{x_1^4}{4} + \frac{x_2^2}{2} - x_1 x_2 + x_1 - x_2.$$

- a. Use MATLAB to plot the level sets of f at levels $-0.72, -0.6, -0.2, 0.5$,
2. Locate the minimizers of f from the plots of the level sets.
- b. Apply the DFP algorithm to minimize the function above with the following starting initial conditions: (i) $[0, 0]^\top$; (ii) $[1.5, 1]^\top$. Use $\mathbf{H}_0 = \mathbf{I}_2$. Does the algorithm converge to the same point for the two initial conditions? If not, explain.

CHAPTER 12

SOLVING LINEAR EQUATIONS

12.1 Least-Squares Analysis

Consider a system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $m \geq n$, and $\text{rank } \mathbf{A} = n$. Note that the number of unknowns, n , is no larger than the number of equations, m . If \mathbf{b} does not belong to the range of \mathbf{A} , that is, if $\mathbf{b} \notin \mathcal{R}(\mathbf{A})$, then this system of equations is said to be *inconsistent* or *overdetermined*. In this case there is no solution to the above set of equations. Our goal then is to find the vector (or vectors) \mathbf{x} minimizing $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$. This problem is a special case of the nonlinear least-squares problem discussed in Section 9.4.

Let \mathbf{x}^* be a vector that minimizes $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$; that is, for all $\mathbf{x} \in \mathbb{R}^n$,

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 \geq \|\mathbf{A}\mathbf{x}^* - \mathbf{b}\|^2.$$

We refer to the vector \mathbf{x}^* as a *least-squares solution* to $\mathbf{A}\mathbf{x} = \mathbf{b}$. In the case where $\mathbf{A}\mathbf{x} = \mathbf{b}$ has a solution, then the solution is a least-squares solution.

Otherwise, a least-squares solution minimizes the norm of the difference between the left- and right-hand sides of the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$. To characterize least-squares solutions, we need the following lemma.

Lemma 12.1 *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \geq n$. Then, $\text{rank } \mathbf{A} = n$ if and only if $\text{rank } \mathbf{A}^\top \mathbf{A} = n$ (i.e., the square matrix $\mathbf{A}^\top \mathbf{A}$ is nonsingular).* \square

Proof. \Rightarrow : Suppose that $\text{rank } \mathbf{A} = n$. To show $\text{rank } \mathbf{A}^\top \mathbf{A} = n$, it is equivalent to show $\mathcal{N}(\mathbf{A}^\top \mathbf{A}) = \{\mathbf{0}\}$. To proceed, let $\mathbf{x} \in \mathcal{N}(\mathbf{A}^\top \mathbf{A})$; that is, $\mathbf{A}^\top \mathbf{A}\mathbf{x} = \mathbf{0}$. Therefore,

$$\|\mathbf{A}\mathbf{x}\|^2 = \mathbf{x}^\top \mathbf{A}^\top \mathbf{A}\mathbf{x} = 0,$$

which implies that $\mathbf{A}\mathbf{x} = \mathbf{0}$. Because $\text{rank } \mathbf{A} = n$, we have $\mathbf{x} = \mathbf{0}$.

\Leftarrow : Suppose that $\text{rank } \mathbf{A}^\top \mathbf{A} = n$; that is, $\mathcal{N}(\mathbf{A}^\top \mathbf{A}) = \{\mathbf{0}\}$. To show $\text{rank } \mathbf{A} = n$, it is equivalent to show that $\mathcal{N}(\mathbf{A}) = \{\mathbf{0}\}$. To proceed, let $\mathbf{x} \in \mathcal{N}(\mathbf{A})$; that is, $\mathbf{A}\mathbf{x} = \mathbf{0}$. Then, $\mathbf{A}^\top \mathbf{A}\mathbf{x} = \mathbf{0}$, and hence $\mathbf{x} = \mathbf{0}$. \blacksquare

Recall that we assume throughout that $\text{rank } \mathbf{A} = n$. By Lemma 12.1 we conclude that $(\mathbf{A}^\top \mathbf{A})^{-1}$ exists. The following theorem characterizes the least-squares solution.

Theorem 12.1 *The unique vector \mathbf{x}^* that minimizes $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$ is given by the solution to the equation $\mathbf{A}^\top \mathbf{A}\mathbf{x} = \mathbf{A}^\top \mathbf{b}$; that is, $\mathbf{x}^* = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$.* \square

Proof. Let $\mathbf{x}^* = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$. First observe that

$$\begin{aligned} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 &= \|\mathbf{A}(\mathbf{x} - \mathbf{x}^*) + (\mathbf{A}\mathbf{x}^* - \mathbf{b})\|^2 \\ &= (\mathbf{A}(\mathbf{x} - \mathbf{x}^*) + (\mathbf{A}\mathbf{x}^* - \mathbf{b}))^\top (\mathbf{A}(\mathbf{x} - \mathbf{x}^*) + (\mathbf{A}\mathbf{x}^* - \mathbf{b})) \\ &= \|\mathbf{A}(\mathbf{x} - \mathbf{x}^*)\|^2 + \|\mathbf{A}\mathbf{x}^* - \mathbf{b}\|^2 + 2[\mathbf{A}(\mathbf{x} - \mathbf{x}^*)]^\top (\mathbf{A}\mathbf{x}^* - \mathbf{b}). \end{aligned}$$

We now show that the last term in this equation is zero. Indeed, substituting the expression above for \mathbf{x}^* ,

$$\begin{aligned} [\mathbf{A}(\mathbf{x} - \mathbf{x}^*)]^\top (\mathbf{A}\mathbf{x}^* - \mathbf{b}) &= (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A}^\top [\mathbf{A}(\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top - \mathbf{I}_n] \mathbf{b} \\ &= (\mathbf{x} - \mathbf{x}^*)^\top [(\mathbf{A}^\top \mathbf{A})(\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top - \mathbf{A}^\top] \mathbf{b} \\ &= (\mathbf{x} - \mathbf{x}^*)^\top (\mathbf{A}^\top - \mathbf{A}^\top) \mathbf{b} \\ &= 0. \end{aligned}$$

Hence,

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 = \|\mathbf{A}(\mathbf{x} - \mathbf{x}^*)\|^2 + \|\mathbf{A}\mathbf{x}^* - \mathbf{b}\|^2.$$

If $\mathbf{x} \neq \mathbf{x}^*$, then $\|\mathbf{A}(\mathbf{x} - \mathbf{x}^*)\|^2 > 0$, because $\text{rank } \mathbf{A} = n$. Thus, if $\mathbf{x} \neq \mathbf{x}^*$, we have

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 > \|\mathbf{A}\mathbf{x}^* - \mathbf{b}\|^2.$$

Thus, $\mathbf{x}^* = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$ is the unique minimizer of $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$. \blacksquare

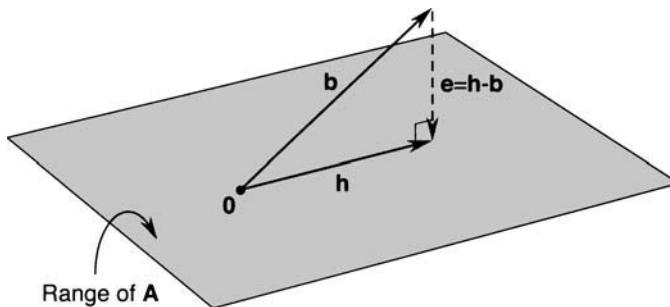


Figure 12.1 Orthogonal projection of \mathbf{b} on the subspace $\mathcal{R}(\mathbf{A})$.

We now give a geometric interpretation of the Theorem 12.1. First note that the columns of \mathbf{A} span the range $\mathcal{R}(\mathbf{A})$ of \mathbf{A} , which is an n -dimensional subspace of \mathbb{R}^m . The equation $\mathbf{Ax} = \mathbf{b}$ has a solution if and only if \mathbf{b} lies in this n -dimensional subspace $\mathcal{R}(\mathbf{A})$. If $m = n$, then $\mathbf{b} \in \mathcal{R}(\mathbf{A})$ always, and the solution is $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$. Suppose now that $m > n$. Intuitively, we would expect the “likelihood” of $\mathbf{b} \in \mathcal{R}(\mathbf{A})$ to be small, because the subspace spanned by the columns of \mathbf{A} is very “thin.” Therefore, let us suppose that \mathbf{b} does not belong to $\mathcal{R}(\mathbf{A})$. We wish to find a point $\mathbf{h} \in \mathcal{R}(\mathbf{A})$ that is “closest” to \mathbf{b} . Geometrically, the point \mathbf{h} should be such that the vector $\mathbf{e} = \mathbf{b} - \mathbf{h}$ is orthogonal to the subspace $\mathcal{R}(\mathbf{A})$ (see Figure 12.1). Recall that a vector $\mathbf{e} \in \mathbb{R}^m$ is said to be orthogonal to the subspace $\mathcal{R}(\mathbf{A})$ if it is orthogonal to every vector in this subspace. We call \mathbf{h} the *orthogonal projection* of \mathbf{b} onto the subspace $\mathcal{R}(\mathbf{A})$. It turns out that $\mathbf{h} = \mathbf{Ax}^* = \mathbf{A}(\mathbf{A}^\top \mathbf{A})^{-1}\mathbf{A}^\top \mathbf{b}$. Hence, the vector $\mathbf{h} \in \mathcal{R}(\mathbf{A})$ minimizing $\|\mathbf{b} - \mathbf{h}\|$ is exactly the orthogonal projection of \mathbf{b} onto $\mathcal{R}(\mathbf{A})$. In other words, the vector \mathbf{x}^* minimizing $\|\mathbf{Ax} - \mathbf{b}\|$ is exactly the vector that makes $\mathbf{Ax} - \mathbf{b}$ orthogonal to $\mathcal{R}(\mathbf{A})$.

To proceed further, we write $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_n]$, where $\mathbf{a}_1, \dots, \mathbf{a}_n$ are the columns of \mathbf{A} . The vector \mathbf{e} is orthogonal to $\mathcal{R}(\mathbf{A})$ if and only if it is orthogonal to each of the columns $\mathbf{a}_1, \dots, \mathbf{a}_n$ of \mathbf{A} . To see this, note that

$$\langle \mathbf{e}, \mathbf{a}_i \rangle = 0, \quad i = 1, \dots, n$$

if and only if for any set of scalars $\{x_1, x_2, \dots, x_n\}$, we also have

$$\langle \mathbf{e}, x_1 \mathbf{a}_1 + \dots + x_n \mathbf{a}_n \rangle = 0.$$

Any vector in $\mathcal{R}(\mathbf{A})$ has the form $x_1 \mathbf{a}_1 + \dots + x_n \mathbf{a}_n$.

Proposition 12.1 Let $\mathbf{h} \in \mathcal{R}(\mathbf{A})$ be such that $\mathbf{h} - \mathbf{b}$ is orthogonal to $\mathcal{R}(\mathbf{A})$. Then, $\mathbf{h} = \mathbf{Ax}^* = \mathbf{A}(\mathbf{A}^\top \mathbf{A})^{-1}\mathbf{A}^\top \mathbf{b}$. \square

Proof. Because $\mathbf{h} \in \mathcal{R}(\mathbf{A}) = \text{span}[\mathbf{a}_1, \dots, \mathbf{a}_n]$, it has the form $\mathbf{h} = x_1 \mathbf{a}_1 + \dots + x_n \mathbf{a}_n$, where $x_1, \dots, x_n \in \mathbb{R}$. To find x_1, \dots, x_n , we use the assumption

that $\mathbf{e} = \mathbf{h} - \mathbf{b}$ is orthogonal to $\text{span}[\mathbf{a}_1, \dots, \mathbf{a}_n]$; that is, for all $i = 1, \dots, n$, we have

$$\langle \mathbf{h} - \mathbf{b}, \mathbf{a}_i \rangle = 0,$$

or, equivalently,

$$\langle \mathbf{h}, \mathbf{a}_i \rangle = \langle \mathbf{b}, \mathbf{a}_i \rangle.$$

Substituting \mathbf{h} into the equations above, we obtain a set of n linear equations of the form

$$\langle \mathbf{a}_1, \mathbf{a}_i \rangle x_1 + \cdots + \langle \mathbf{a}_n, \mathbf{a}_i \rangle x_n = \langle \mathbf{b}, \mathbf{a}_i \rangle, \quad i = 1, \dots, n.$$

In matrix notation this system of n equations can be represented as

$$\begin{bmatrix} \langle \mathbf{a}_1, \mathbf{a}_1 \rangle & \cdots & \langle \mathbf{a}_n, \mathbf{a}_1 \rangle \\ \vdots & & \vdots \\ \langle \mathbf{a}_1, \mathbf{a}_n \rangle & \cdots & \langle \mathbf{a}_n, \mathbf{a}_n \rangle \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \langle \mathbf{b}, \mathbf{a}_1 \rangle \\ \vdots \\ \langle \mathbf{b}, \mathbf{a}_n \rangle \end{bmatrix}.$$

Note that we can write

$$\begin{bmatrix} \langle \mathbf{a}_1, \mathbf{a}_1 \rangle & \cdots & \langle \mathbf{a}_n, \mathbf{a}_1 \rangle \\ \vdots & & \vdots \\ \langle \mathbf{a}_1, \mathbf{a}_n \rangle & \cdots & \langle \mathbf{a}_n, \mathbf{a}_n \rangle \end{bmatrix} = \mathbf{A}^\top \mathbf{A} = \begin{bmatrix} \mathbf{a}_1^\top \\ \vdots \\ \mathbf{a}_n^\top \end{bmatrix} [\mathbf{a}_1 \ \cdots \ \mathbf{a}_n].$$

We also note that

$$\begin{bmatrix} \langle \mathbf{b}, \mathbf{a}_1 \rangle \\ \vdots \\ \langle \mathbf{b}, \mathbf{a}_n \rangle \end{bmatrix} = \mathbf{A}^\top \mathbf{b} = \begin{bmatrix} \mathbf{a}_1^\top \\ \vdots \\ \mathbf{a}_n^\top \end{bmatrix} \mathbf{b}.$$

Because $\text{rank } \mathbf{A} = n$, $\mathbf{A}^\top \mathbf{A}$ is nonsingular, and thus we conclude that

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b} = \mathbf{x}^*.$$

■

Notice that the matrix

$$\mathbf{A}^\top \mathbf{A} = \begin{bmatrix} \langle \mathbf{a}_1, \mathbf{a}_1 \rangle & \cdots & \langle \mathbf{a}_n, \mathbf{a}_1 \rangle \\ \vdots & & \vdots \\ \langle \mathbf{a}_1, \mathbf{a}_n \rangle & \cdots & \langle \mathbf{a}_n, \mathbf{a}_n \rangle \end{bmatrix}$$

plays an important role in the least-squares solution. This matrix is often called the *Gram matrix* (or *Gramian*).

An alternative method of arriving at the least-squares solution is to proceed as follows. First, we write

$$\begin{aligned} f(\mathbf{x}) &= \|\mathbf{Ax} - \mathbf{b}\|^2 \\ &= (\mathbf{Ax} - \mathbf{b})^\top (\mathbf{Ax} - \mathbf{b}) \\ &= \frac{1}{2} \mathbf{x}^\top (2\mathbf{A}^\top \mathbf{A}) \mathbf{x} - \mathbf{x}^\top (2\mathbf{A}^\top \mathbf{b}) + \mathbf{b}^\top \mathbf{b}. \end{aligned}$$

Therefore, f is a quadratic function. The quadratic term is positive definite because $\text{rank } \mathbf{A} = n$. Thus, the unique minimizer of f is obtained by solving the FONC (see Exercise 6.33); that is,

$$\nabla f(\mathbf{x}) = 2\mathbf{A}^\top \mathbf{Ax} - 2\mathbf{A}^\top \mathbf{b} = \mathbf{0}.$$

The only solution to the equation $\nabla f(\mathbf{x}) = \mathbf{0}$ is $\mathbf{x}^* = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$.

Example 12.1 Suppose that you are given two different types of concrete. The first type contains 30% cement, 40% gravel, and 30% sand (all percentages of weight). The second type contains 10% cement, 20% gravel, and 70% sand. How many pounds of each type of concrete should you mix together so that you get a concrete mixture that has as close as possible to a total of 5 pounds of cement, 3 pounds of gravel, and 4 pounds of sand?

The problem can be formulated as a least-squares problem with

$$\mathbf{A} = \begin{bmatrix} 0.3 & 0.1 \\ 0.4 & 0.2 \\ 0.3 & 0.7 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 5 \\ 3 \\ 4 \end{bmatrix},$$

where the decision variable is $\mathbf{x} = [x_1, x_2]^\top$ and x_1 and x_2 are the amounts of concrete of the first and second types, respectively. After some algebra, we obtain the solution:

$$\begin{aligned} \mathbf{x}^* &= (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b} \\ &= \frac{1}{(0.34)(0.54) - (0.32)^2} \begin{bmatrix} 0.54 & -0.32 \\ -0.32 & 0.34 \end{bmatrix} \begin{bmatrix} 3.9 \\ 3.9 \end{bmatrix} \\ &= \begin{bmatrix} 10.6 \\ 0.961 \end{bmatrix}. \end{aligned}$$

(For a variation of this problem solved using a different method, see Example 15.7.) ■

We now give an example in which least-squares analysis is used to fit measurements by a straight line.

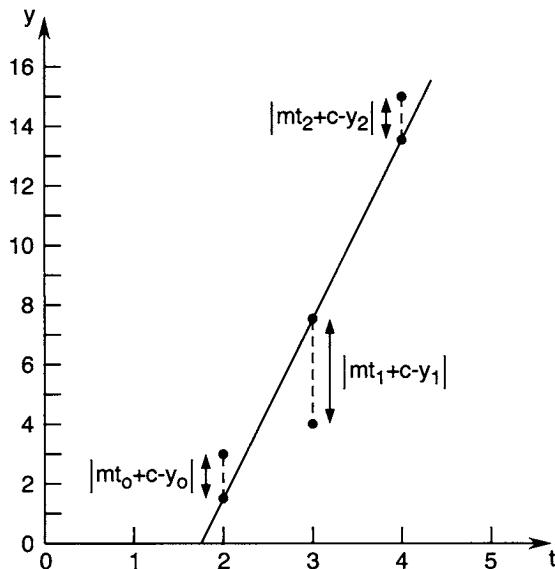
Table 12.1 Experimental Data for Example 12.2.

i	0	1	2
t_i	2	3	4
y_i	3	4	15

Example 12.2 Line Fitting. Suppose that a process has a single input $t \in \mathbb{R}$ and a single output $y \in \mathbb{R}$. Suppose that we perform an experiment on the process, resulting in a number of measurements, as displayed in Table 12.1. The i th measurement results in the input labeled t_i and the output labeled y_i . We would like to find a straight line given by

$$y = mt + c$$

that fits the experimental data. In other words, we wish to find two numbers, m and c , such that $y_i = mt_i + c$, $i = 0, 1, 2$. However, it is apparent that there is no choice of m and c that results in the requirement above; that is, there is no straight line that passes through all three points simultaneously. Therefore, we would like to find the values of m and c that best fit the data. A graphical illustration of our problem is shown in Figure 12.2.

**Figure 12.2** Fitting a straight line to experimental data.

We can represent our problem as a system of three linear equations of the form

$$\begin{aligned} 2m + c &= 3 \\ 3m + c &= 4 \\ 4m + c &= 15. \end{aligned}$$

We can write this system of equations as

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 3 & 1 \\ 4 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 3 \\ 4 \\ 15 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} m \\ c \end{bmatrix}.$$

Note that since

$$\text{rank } \mathbf{A} < \text{rank } [\mathbf{A}, \mathbf{b}],$$

the vector \mathbf{b} does not belong to the range of \mathbf{A} . Thus, as we have noted before, the system of equations above is inconsistent.

The straight line of best fit is the one that minimizes

$$\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 = \sum_{i=0}^2 (mt_i + c - y_i)^2.$$

Therefore, our problem lies in the class of least-squares problems. Note that the foregoing function of m and c is simply the total squared vertical distance (squared error) between the straight line defined by m and c and the experimental points. The solution to our least-squares problem is

$$\mathbf{x}^* = \begin{bmatrix} m^* \\ c^* \end{bmatrix} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b} = \begin{bmatrix} 6 \\ -32/3 \end{bmatrix}.$$

Note that the error vector $\mathbf{e} = \mathbf{A}\mathbf{x}^* - \mathbf{b}$ is orthogonal to each column of \mathbf{A} . ■

Next, we give an example of the use of least-squares in wireless communications.

Example 12.3 Attenuation Estimation. A wireless transmitter sends a discrete-time signal $\{s_0, s_1, s_2\}$ (of duration 3) to a receiver, as shown in Figure 12.3. The real number s_i is the value of the signal at time i .

The transmitted signal takes two paths to the receiver: a direct path, with delay 10 and attenuation factor a_1 , and an indirect (reflected) path, with delay 12 and attenuation factor a_2 . The received signal is the sum of the signals from these two paths, with their respective delays and attenuation factors.

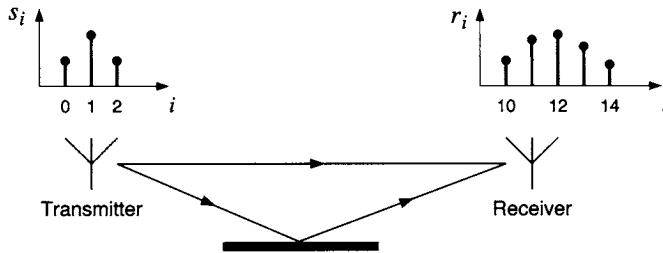


Figure 12.3 Wireless transmission in Example 12.3.

Suppose that the received signal is measured from times 10 through 14 as $r_{10}, r_{11}, \dots, r_{14}$, as shown in the figure. We wish to compute the least-squares estimates of a_1 and a_2 , based on the following values:

$$\frac{s_0 \quad s_1 \quad s_2 \quad r_{10} \quad r_{11} \quad r_{12} \quad r_{13} \quad r_{14}}{1 \quad 2 \quad 1 \quad 4 \quad 7 \quad 8 \quad 6 \quad 3}.$$

The problem can be posed as a least-squares problem with

$$\mathbf{A} = \begin{bmatrix} s_0 & 0 \\ s_1 & 0 \\ s_2 & s_0 \\ 0 & s_1 \\ 0 & s_2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} r_{10} \\ r_{11} \\ r_{12} \\ r_{13} \\ r_{14} \end{bmatrix}.$$

The least-squares estimate is given by

$$\begin{aligned} \begin{bmatrix} a_1^* \\ a_2^* \end{bmatrix} &= (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b} \\ &= \begin{bmatrix} \|\mathbf{s}\|^2 & s_0 s_2 \\ s_0 s_2 & \|\mathbf{s}\|^2 \end{bmatrix}^{-1} \begin{bmatrix} s_0 r_{10} + s_1 r_{11} + s_2 r_{12} \\ s_0 r_{12} + s_1 r_{13} + s_2 r_{14} \end{bmatrix} \\ &= \begin{bmatrix} 6 & 1 \\ 1 & 6 \end{bmatrix}^{-1} \begin{bmatrix} 4 + 14 + 8 \\ 8 + 12 + 3 \end{bmatrix} \\ &= \frac{1}{35} \begin{bmatrix} 6 & -1 \\ -1 & 6 \end{bmatrix} \begin{bmatrix} 26 \\ 23 \end{bmatrix} \\ &= \frac{1}{35} \begin{bmatrix} 133 \\ 112 \end{bmatrix}. \end{aligned}$$

■

We now give a simple example where the least-squares method is used in digital signal processing.

Example 12.4 Discrete Fourier Series. Suppose that we are given a discrete-time signal, represented by the vector

$$\mathbf{b} = [b_1, b_2, \dots, b_m]^\top.$$

We wish to approximate this signal by a sum of sinusoids. Specifically, we approximate \mathbf{b} by the vector

$$y_0 \mathbf{c}^{(0)} + \sum_{k=1}^n \left(y_k \mathbf{c}^{(k)} + z_k \mathbf{s}^{(k)} \right),$$

where $y_0, y_1, \dots, y_n, z_1, \dots, z_n \in \mathbb{R}$ and the vectors $\mathbf{c}^{(k)}$ and $\mathbf{s}^{(k)}$ are given by

$$\begin{aligned} \mathbf{c}^{(0)} &= \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \dots, \frac{1}{\sqrt{2}} \right]^\top, \\ \mathbf{c}^{(k)} &= \left[\cos \left(1 \frac{2k\pi}{m} \right), \cos \left(2 \frac{2k\pi}{m} \right), \dots, \cos \left(m \frac{2k\pi}{m} \right) \right]^\top, \quad k = 1, \dots, n, \\ \mathbf{s}^{(k)} &= \left[\sin \left(1 \frac{2k\pi}{m} \right), \sin \left(2 \frac{2k\pi}{m} \right), \dots, \sin \left(m \frac{2k\pi}{m} \right) \right]^\top, \quad k = 1, \dots, n. \end{aligned}$$

We call the sum of sinusoids above a *discrete Fourier series* (although, strictly speaking, it is not a series but a finite sum). We wish to find $y_0, y_1, \dots, y_n, z_1, \dots, z_n$ such that

$$\left\| \left(y_0 \mathbf{c}^{(0)} + \sum_{k=1}^n y_k \mathbf{c}^{(k)} + z_k \mathbf{s}^{(k)} \right) - \mathbf{b} \right\|^2$$

is minimized.

To proceed, we define

$$\begin{aligned} \mathbf{A} &= \left[\mathbf{c}^{(0)}, \mathbf{c}^{(1)}, \dots, \mathbf{c}^{(n)}, \mathbf{s}^{(1)}, \dots, \mathbf{s}^{(n)} \right], \\ \mathbf{x} &= [y_0, y_1, \dots, y_n, z_1, \dots, z_n]^\top. \end{aligned}$$

Our problem can be reformulated as minimizing

$$\|\mathbf{Ax} - \mathbf{b}\|^2.$$

We assume that $m \geq 2n+1$. To find the solution, we first compute $\mathbf{A}^\top \mathbf{A}$. We make use of the following trigonometric identities: For any nonzero integer k that is not an integral multiple of m , we have

$$\sum_{i=1}^m \cos \left(i \frac{2k\pi}{m} \right) = 0,$$

$$\sum_{i=1}^m \sin \left(i \frac{2k\pi}{m} \right) = 0.$$

With the aid of these identities, we can verify that

$$\begin{aligned}\mathbf{c}^{(k)\top} \mathbf{c}^{(j)} &= \begin{cases} m/2 & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \\ \mathbf{s}^{(k)\top} \mathbf{s}^{(j)} &= \begin{cases} m/2 & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \\ \mathbf{c}^{(k)\top} \mathbf{s}^{(j)} &= 0 \quad \text{for any } k, j.\end{aligned}$$

Hence,

$$\mathbf{A}^\top \mathbf{A} = \frac{m}{2} \mathbf{I}_{2n+1},$$

which is clearly nonsingular, with inverse

$$(\mathbf{A}^\top \mathbf{A})^{-1} = \frac{2}{m} \mathbf{I}_{2n+1}.$$

Therefore, the solution to our problem is

$$\begin{aligned}\mathbf{x}^* &= [y_0^*, y_1^*, \dots, y_n^*, z_1^*, \dots, z_n^*]^\top \\ &= (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b} \\ &= \frac{2}{m} \mathbf{A}^\top \mathbf{b}.\end{aligned}$$

We represent the solution as

$$\begin{aligned}y_0^* &= \frac{\sqrt{2}}{m} \sum_{i=1}^m b_i, \\ y_k^* &= \frac{2}{m} \sum_{i=1}^m b_i \cos\left(i \frac{2k\pi}{m}\right), \quad k = 1, \dots, n, \\ z_k^* &= \frac{2}{m} \sum_{i=1}^m b_i \sin\left(i \frac{2k\pi}{m}\right), \quad k = 1, \dots, n.\end{aligned}$$

We call these *discrete Fourier coefficients*. ■

Finally, we show how least-squares analysis can be used to derive formulas for orthogonal projectors.

Example 12.5 *Orthogonal Projectors.* Let $\mathcal{V} \subset \mathbb{R}^n$ be a subspace. Given a vector $\mathbf{x} \in \mathbb{R}^n$, we write the orthogonal decomposition of \mathbf{x} as

$$\mathbf{x} = \mathbf{x}_\mathcal{V} + \mathbf{x}_{\mathcal{V}^\perp},$$

where $\mathbf{x}_\mathcal{V} \in \mathcal{V}$ is the orthogonal projection of \mathbf{x} onto \mathcal{V} and $\mathbf{x}_{\mathcal{V}^\perp} \in \mathcal{V}^\perp$ is the orthogonal projection of \mathbf{x} onto \mathcal{V}^\perp . (See Section 3.3; also recall that \mathcal{V}^\perp is

the orthogonal complement of \mathcal{V} .) We can write $\mathbf{x}_\mathcal{V} = \mathbf{P}\mathbf{x}$ for some matrix \mathbf{P} called the *orthogonal projector*. In the following, we derive expressions for \mathbf{P} for the case where $\mathcal{V} = \mathcal{R}(\mathbf{A})$ and the case where $\mathcal{V} = \mathcal{N}(\mathbf{A})$.

Consider a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \geq n$, and $\text{rank } \mathbf{A} = n$. Let $\mathcal{V} = \mathcal{R}(\mathbf{A})$ be the range of \mathbf{A} (note that any subspace can be written as the range of some matrix). In this case we can write an expression for \mathbf{P} in terms of \mathbf{A} , as follows. By Proposition 12.1 we have $\mathbf{x}_\mathcal{V} = \mathbf{A}(\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{x}$, whence $\mathbf{P} = \mathbf{A}(\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top$. Note that by Proposition 12.1, we may also write

$$\mathbf{x}_\mathcal{V} = \arg \min_{\mathbf{y} \in \mathcal{V}} \|\mathbf{y} - \mathbf{x}\|.$$

Next, consider a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \leq n$, and $\text{rank } \mathbf{A} = m$. Let $\mathcal{V} = \mathcal{N}(\mathbf{A})$ be the nullspace of \mathbf{A} (note that any subspace can be written as the nullspace of some matrix). To derive an expression for the orthogonal projector \mathbf{P} in terms of \mathbf{A} for this case, we use the formula derived above and the identity $\mathcal{N}(\mathbf{A})^\perp = \mathcal{R}(\mathbf{A}^\top)$ (see Theorem 3.4). Indeed, if $\mathcal{U} = \mathcal{R}(\mathbf{A}^\top)$, then the orthogonal decomposition with respect to \mathcal{U} is $\mathbf{x} = \mathbf{x}_\mathcal{U} + \mathbf{x}_{\mathcal{U}^\perp}$, where $\mathbf{x}_\mathcal{U} = \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{A}\mathbf{x}$ (using the formula derived above). Because $\mathcal{N}(\mathbf{A})^\perp = \mathcal{R}(\mathbf{A}^\top)$, we deduce that $\mathbf{x}_{\mathcal{V}^\perp} = \mathbf{x}_\mathcal{U} = \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{A}\mathbf{x}$. Hence,

$$\mathbf{x}_\mathcal{V} = \mathbf{x} - \mathbf{x}_{\mathcal{V}^\perp} = \mathbf{x} - \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{A}\mathbf{x} = (\mathbf{I} - \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{A})\mathbf{x}.$$

Thus, the orthogonal projector in this case is $\mathbf{P} = \mathbf{I} - \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{A}$. ■

12.2 The Recursive Least-Squares Algorithm

Consider again the example in Section 12.1. We are given experimental points (t_0, y_0) , (t_1, y_1) , and (t_2, y_2) , and we find the parameters m^* and c^* of the straight line that best fits these data in the least-squares sense. Suppose that we are now given an extra measurement point (t_3, y_3) , so that we now have a set of four experimental data points: (t_0, y_0) , (t_1, y_1) , (t_2, y_2) , and (t_3, y_3) . We can similarly go through the procedure for finding the parameters of the line of best fit for this set of four points. However, as we shall see, there is a more efficient way: We can use previous calculations of m^* and c^* for the three data points to calculate the parameters for the four data points. In effect, we simply “update” our values of m^* and c^* to accommodate the new data point. This procedure, called the *recursive least-squares (RLS) algorithm*, is the topic of this section.

To derive the RLS algorithm, first consider the problem of minimizing $\|\mathbf{A}_0 \mathbf{x} - \mathbf{b}^{(0)}\|^2$. We know that the solution to this is given by $\mathbf{x}^{(0)} = \mathbf{G}_0^{-1} \mathbf{A}_0^\top \mathbf{b}^{(0)}$, where $\mathbf{G}_0 = \mathbf{A}_0^\top \mathbf{A}_0$. Suppose now that we are given more data, in the form of a matrix \mathbf{A}_1 and a vector $\mathbf{b}^{(1)}$. Consider now the problem of minimizing

$$\left\| \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b}^{(0)} \\ \mathbf{b}^{(1)} \end{bmatrix} \right\|^2.$$

The solution is given by

$$\mathbf{x}^{(1)} = \mathbf{G}_1^{-1} \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{b}^{(0)} \\ \mathbf{b}^{(1)} \end{bmatrix},$$

where

$$\mathbf{G}_1 = \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \end{bmatrix}.$$

Our goal is to write $\mathbf{x}^{(1)}$ as a function of $\mathbf{x}^{(0)}$, \mathbf{G}_0 , and the new data \mathbf{A}_1 and $\mathbf{b}^{(1)}$. To this end, we first write \mathbf{G}_1 as

$$\begin{aligned} \mathbf{G}_1 &= [\mathbf{A}_0^\top \ \mathbf{A}_1^\top] \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \end{bmatrix} \\ &= \mathbf{A}_0^\top \mathbf{A}_0 + \mathbf{A}_1^\top \mathbf{A}_1 \\ &= \mathbf{G}_0 + \mathbf{A}_1^\top \mathbf{A}_1. \end{aligned}$$

Next, we write

$$\begin{aligned} \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{b}^{(0)} \\ \mathbf{b}^{(1)} \end{bmatrix} &= [\mathbf{A}_0^\top \ \mathbf{A}_1^\top] \begin{bmatrix} \mathbf{b}^{(0)} \\ \mathbf{b}^{(1)} \end{bmatrix} \\ &= \mathbf{A}_0^\top \mathbf{b}^{(0)} + \mathbf{A}_1^\top \mathbf{b}^{(1)}. \end{aligned}$$

To proceed further, we write $\mathbf{A}_0^\top \mathbf{b}^{(0)}$ as

$$\begin{aligned} \mathbf{A}_0^\top \mathbf{b}^{(0)} &= \mathbf{G}_0 \mathbf{G}_0^{-1} \mathbf{A}_0^\top \mathbf{b}^{(0)} \\ &= \mathbf{G}_0 \mathbf{x}^{(0)} \\ &= (\mathbf{G}_1 - \mathbf{A}_1^\top \mathbf{A}_1) \mathbf{x}^{(0)} \\ &= \mathbf{G}_1 \mathbf{x}^{(0)} - \mathbf{A}_1^\top \mathbf{A}_1 \mathbf{x}^{(0)}. \end{aligned}$$

Combining these formulas, we see that we can write $\mathbf{x}^{(1)}$ as

$$\begin{aligned} \mathbf{x}^{(1)} &= \mathbf{G}_1^{-1} \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{b}^{(0)} \\ \mathbf{b}^{(1)} \end{bmatrix} \\ &= \mathbf{G}_1^{-1} (\mathbf{G}_1 \mathbf{x}^{(0)} - \mathbf{A}_1^\top \mathbf{A}_1 \mathbf{x}^{(0)} + \mathbf{A}_1^\top \mathbf{b}^{(1)}) \\ &= \mathbf{x}^{(0)} + \mathbf{G}_1^{-1} \mathbf{A}_1^\top (\mathbf{b}^{(1)} - \mathbf{A}_1 \mathbf{x}^{(0)}), \end{aligned}$$

where \mathbf{G}_1 can be calculated using

$$\mathbf{G}_1 = \mathbf{G}_0 + \mathbf{A}_1^\top \mathbf{A}_1.$$

We note that with this formula, $\mathbf{x}^{(1)}$ can be computed using only $\mathbf{x}^{(0)}$, \mathbf{A}_1 , $\mathbf{b}^{(1)}$, and \mathbf{G}_0 . Hence, we have a way of using our previous efforts in calculating $\mathbf{x}^{(0)}$ to compute $\mathbf{x}^{(1)}$, without having to compute $\mathbf{x}^{(1)}$ directly from scratch. The solution $\mathbf{x}^{(1)}$ is obtained from $\mathbf{x}^{(0)}$ by a simple update equation that adds to $\mathbf{x}^{(0)}$ a “correction term” $\mathbf{G}_1^{-1}\mathbf{A}_1^\top (\mathbf{b}^{(1)} - \mathbf{A}_1\mathbf{x}^{(0)})$. Observe that if the new data are consistent with the old data, that is, $\mathbf{A}_1\mathbf{x}^{(0)} = \mathbf{b}^{(1)}$, then the correction term is 0 and the updated solution $\mathbf{x}^{(1)}$ is equal to the previous solution $\mathbf{x}^{(0)}$.

We can generalize the argument above to write a recursive algorithm for updating the least-squares solution as new data arrive. At the $(k+1)$ th iteration, we have

$$\begin{aligned}\mathbf{G}_{k+1} &= \mathbf{G}_k + \mathbf{A}_{k+1}^\top \mathbf{A}_{k+1} \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \mathbf{G}_{k+1}^{-1} \mathbf{A}_{k+1}^\top (\mathbf{b}^{(k+1)} - \mathbf{A}_{k+1}\mathbf{x}^{(k)}).\end{aligned}$$

The vector $\mathbf{b}^{(k+1)} - \mathbf{A}_{k+1}\mathbf{x}^{(k)}$ is often called the *innovation*. As before, observe that if the innovation is zero, then the updated solution $\mathbf{x}^{(k+1)}$ is equal to the previous solution $\mathbf{x}^{(k)}$.

We can see from the above that to compute $\mathbf{x}^{(k+1)}$ from $\mathbf{x}^{(k)}$ we need \mathbf{G}_{k+1}^{-1} , rather than \mathbf{G}_{k+1} . It turns out that we can derive an update formula for \mathbf{G}_{k+1}^{-1} itself. To do so, we need the following technical lemma, which is a generalization of the Sherman-Morrison formula (Lemma 11.1), due to Woodbury, and hence also called the *Sherman-Morrison-Woodbury formula* (see [63, p. 124] or [53, p. 50]).

Lemma 12.2 *Let \mathbf{A} be a nonsingular matrix. Let \mathbf{U} and \mathbf{V} be matrices such that $\mathbf{I} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U}$ is nonsingular. Then, $\mathbf{A} + \mathbf{U}\mathbf{V}$ is nonsingular, and*

$$(\mathbf{A} + \mathbf{U}\mathbf{V})^{-1} = \mathbf{A}^{-1} - (\mathbf{A}^{-1}\mathbf{U})(\mathbf{I} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}(\mathbf{V}\mathbf{A}^{-1}).$$

□

Proof. We can prove the result easily by verification. ■

Using Lemma 12.2 we get

$$\begin{aligned}\mathbf{G}_{k+1}^{-1} &= \left(\mathbf{G}_k + \mathbf{A}_{k+1}^\top \mathbf{A}_{k+1} \right)^{-1} \\ &= \mathbf{G}_k^{-1} - \mathbf{G}_k^{-1} \mathbf{A}_{k+1}^\top (\mathbf{I} + \mathbf{A}_{k+1}\mathbf{G}_k^{-1}\mathbf{A}_{k+1}^\top)^{-1} \mathbf{A}_{k+1}\mathbf{G}_k^{-1}.\end{aligned}$$

For simplicity of notation, we rewrite \mathbf{G}_k^{-1} as \mathbf{P}_k .

We summarize by writing the RLS algorithm using \mathbf{P}_k :

$$\begin{aligned}\mathbf{P}_{k+1} &= \mathbf{P}_k - \mathbf{P}_k \mathbf{A}_{k+1}^\top (\mathbf{I} + \mathbf{A}_{k+1}\mathbf{P}_k\mathbf{A}_{k+1}^\top)^{-1} \mathbf{A}_{k+1}\mathbf{P}_k, \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \mathbf{P}_{k+1} \mathbf{A}_{k+1}^\top (\mathbf{b}^{(k+1)} - \mathbf{A}_{k+1}\mathbf{x}^{(k)}).\end{aligned}$$

In the special case where the new data at each step are such that \mathbf{A}_{k+1} is a matrix consisting of a single row, $\mathbf{A}_{k+1} = \mathbf{a}_{k+1}^\top$, and $\mathbf{b}^{(k+1)}$ is a scalar, $\mathbf{b}^{(k+1)} = b_{k+1}$, we get

$$\begin{aligned}\mathbf{P}_{k+1} &= \mathbf{P}_k - \frac{\mathbf{P}_k \mathbf{a}_{k+1} \mathbf{a}_{k+1}^\top \mathbf{P}_k}{1 + \mathbf{a}_{k+1}^\top \mathbf{P}_k \mathbf{a}_{k+1}}, \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \mathbf{P}_{k+1} \mathbf{a}_{k+1} \left(b_{k+1} - \mathbf{a}_{k+1}^\top \mathbf{x}^{(k)} \right).\end{aligned}$$

Example 12.6 Let

$$\begin{aligned}\mathbf{A}_0 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}, & \mathbf{b}^{(0)} &= \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \\ \mathbf{A}_1 &= \mathbf{a}_1^\top = [2 \ 1], & \mathbf{b}^{(1)} &= b_1 = [3], \\ \mathbf{A}_2 &= \mathbf{a}_2^\top = [3 \ 1], & \mathbf{b}^{(2)} &= b_2 = [4].\end{aligned}$$

First compute the vector $\mathbf{x}^{(0)}$ minimizing $\|\mathbf{A}_0 \mathbf{x} - \mathbf{b}^{(0)}\|^2$. Then, use the RLS algorithm to find $\mathbf{x}^{(2)}$ minimizing

$$\left\| \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b}^{(0)} \\ \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \end{bmatrix} \right\|^2.$$

We have

$$\begin{aligned}\mathbf{P}_0 &= (\mathbf{A}_0^\top \mathbf{A}_0)^{-1} = \begin{bmatrix} 2/3 & -1/3 \\ -1/3 & 2/3 \end{bmatrix}, \\ \mathbf{x}^{(0)} &= \mathbf{P}_0 \mathbf{A}_0^\top \mathbf{b}^{(0)} = \begin{bmatrix} 2/3 \\ 2/3 \end{bmatrix}.\end{aligned}$$

Applying the RLS algorithm twice, we get

$$\begin{aligned}\mathbf{P}_1 &= \mathbf{P}_0 - \frac{\mathbf{P}_0 \mathbf{a}_1 \mathbf{a}_1^\top \mathbf{P}_0}{1 + \mathbf{a}_1^\top \mathbf{P}_0 \mathbf{a}_1} = \begin{bmatrix} 1/3 & -1/3 \\ -1/3 & 2/3 \end{bmatrix}, \\ \mathbf{x}^{(1)} &= \mathbf{x}^{(0)} + \mathbf{P}_1 \mathbf{a}_1 \left(b_1 - \mathbf{a}_1^\top \mathbf{x}^{(0)} \right) = \begin{bmatrix} 1 \\ 2/3 \end{bmatrix}, \\ \mathbf{P}_2 &= \mathbf{P}_1 - \frac{\mathbf{P}_1 \mathbf{a}_2 \mathbf{a}_2^\top \mathbf{P}_1}{1 + \mathbf{a}_2^\top \mathbf{P}_1 \mathbf{a}_2} = \begin{bmatrix} 1/6 & -1/4 \\ -1/4 & 5/8 \end{bmatrix}, \\ \mathbf{x}^{(2)} &= \mathbf{x}^{(1)} + \mathbf{P}_2 \mathbf{a}_2 \left(b_2 - \mathbf{a}_2^\top \mathbf{x}^{(1)} \right) = \begin{bmatrix} 13/12 \\ 5/8 \end{bmatrix}.\end{aligned}$$

We can easily check our solution by computing $\mathbf{x}^{(2)}$ directly using the formula $\mathbf{x}^{(2)} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{b}$, where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}^{(0)} \\ \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \end{bmatrix}.$$

■

12.3 Solution to a Linear Equation with Minimum Norm

Consider now a system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $m \leq n$, and $\text{rank } \mathbf{A} = m$. Note that the number of equations is no larger than the number of unknowns. There may exist an infinite number of solutions to this system of equations. However, as we shall see, there is only one solution that is closest to the origin: the solution to $\mathbf{A}\mathbf{x} = \mathbf{b}$ whose norm $\|\mathbf{x}\|$ is minimal. Let \mathbf{x}^* be this solution; that is, $\mathbf{A}\mathbf{x}^* = \mathbf{b}$ and $\|\mathbf{x}^*\| \leq \|\mathbf{x}\|$ for any \mathbf{x} such that $\mathbf{A}\mathbf{x} = \mathbf{b}$. In other words, \mathbf{x}^* is the solution to the problem

$$\begin{aligned} &\text{minimize} && \|\mathbf{x}\| \\ &\text{subject to} && \mathbf{A}\mathbf{x} = \mathbf{b}. \end{aligned}$$

In Part IV, we study problems of this type in more detail.

Theorem 12.2 *The unique solution \mathbf{x}^* to $\mathbf{A}\mathbf{x} = \mathbf{b}$ that minimizes the norm $\|\mathbf{x}\|$ is given by*

$$\mathbf{x}^* = \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{b}.$$

□

Proof. Let $\mathbf{x}^* = \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{b}$. Note that

$$\begin{aligned} \|\mathbf{x}\|^2 &= \|(\mathbf{x} - \mathbf{x}^*) + \mathbf{x}^*\|^2 \\ &= ((\mathbf{x} - \mathbf{x}^*) + \mathbf{x}^*)^\top ((\mathbf{x} - \mathbf{x}^*) + \mathbf{x}^*) \\ &= \|\mathbf{x} - \mathbf{x}^*\|^2 + \|\mathbf{x}^*\|^2 + 2\mathbf{x}^{*\top}(\mathbf{x} - \mathbf{x}^*). \end{aligned}$$

We now show that

$$\mathbf{x}^{*\top}(\mathbf{x} - \mathbf{x}^*) = 0.$$

Indeed,

$$\begin{aligned} \mathbf{x}^{*\top}(\mathbf{x} - \mathbf{x}^*) &= [\mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{b}]^\top [\mathbf{x} - \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{b}] \\ &= \mathbf{b}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} [\mathbf{A}\mathbf{x} - (\mathbf{A}\mathbf{A}^\top)(\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{b}] \\ &= \mathbf{b}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} [\mathbf{b} - \mathbf{b}] = 0. \end{aligned}$$

Therefore,

$$\|\mathbf{x}\|^2 = \|\mathbf{x}^*\|^2 + \|\mathbf{x} - \mathbf{x}^*\|^2.$$

Because $\|\mathbf{x} - \mathbf{x}^*\|^2 > 0$ for all $\mathbf{x} \neq \mathbf{x}^*$, it follows that for all $\mathbf{x} \neq \mathbf{x}^*$,

$$\|\mathbf{x}\|^2 > \|\mathbf{x}^*\|^2,$$

which implies that

$$\|\mathbf{x}\| > \|\mathbf{x}^*\|. \quad \blacksquare$$

Example 12.7 Find the point closest to the origin of \mathbb{R}^3 on the line of intersection of the two planes defined by the following two equations:

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 1, \\ 4x_1 + x_2 + 3x_3 &= 0. \end{aligned}$$

Note that this problem is equivalent to the problem

$$\begin{aligned} &\text{minimize} \quad \|\mathbf{x}\| \\ &\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \end{aligned}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & -1 \\ 4 & 1 & 3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Thus, the solution to the problem is

$$\mathbf{x}^* = \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{b} = \begin{bmatrix} 0.0952 \\ 0.3333 \\ -0.2381 \end{bmatrix}. \quad \blacksquare$$

In the next section we discuss an iterative algorithm for solving $\mathbf{A}\mathbf{x} = \mathbf{b}$.

12.4 Kaczmarz's Algorithm

As in Section 12.3, let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $m \leq n$, and $\text{rank } \mathbf{A} = m$. We now discuss an iterative algorithm for solving $\mathbf{A}\mathbf{x} = \mathbf{b}$, originally analyzed by Kaczmarz in 1937 [70]. The algorithm converges to the vector $\mathbf{x}^* = \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{b}$ without explicitly having to invert the matrix $\mathbf{A}\mathbf{A}^\top$. This is important from a practical point of view, especially when \mathbf{A} has many rows.

Let \mathbf{a}_j^\top denote the j th row of \mathbf{A} , and b_j the j th component of \mathbf{b} , and μ a positive scalar, $0 < \mu < 2$. With this notation, Kaczmarz's algorithm is:

1. Set $i := 0$, initial condition $\mathbf{x}^{(0)}$.
2. For $j = 1, \dots, m$, set

$$\mathbf{x}^{(im+j)} = \mathbf{x}^{(im+j-1)} + \mu (b_j - \mathbf{a}_j^\top \mathbf{x}^{(im+j-1)}) \frac{\mathbf{a}_j}{\mathbf{a}_j^\top \mathbf{a}_j}.$$
3. Set $i := i + 1$; go to step 2.

In words, Kaczmarz's algorithm works as follows. For the first m iterations ($k = 0, \dots, m-1$), we have

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mu (b_{k+1} - \mathbf{a}_{k+1}^\top \mathbf{x}^{(k)}) \frac{\mathbf{a}_{k+1}}{\mathbf{a}_{k+1}^\top \mathbf{a}_{k+1}},$$

where, in each iteration, we use rows of \mathbf{A} and corresponding components of \mathbf{b} successively. For the $(m+1)$ th iteration, we revert back to the first row of \mathbf{A} and the first component of \mathbf{b} ; that is,

$$\mathbf{x}^{(m+1)} = \mathbf{x}^{(m)} + \mu (b_1 - \mathbf{a}_1^\top \mathbf{x}^{(m)}) \frac{\mathbf{a}_1}{\mathbf{a}_1^\top \mathbf{a}_1}.$$

We continue with the $(m+2)$ th iteration using the second row of \mathbf{A} and second component of \mathbf{b} , and so on, repeating the cycle every m iteration. We can view the scalar μ as the step size of the algorithm. The reason for requiring that μ be between 0 and 2 will become apparent from the convergence analysis.

We now prove the convergence of Kaczmarz's algorithm, using ideas from Kaczmarz's original paper [70] and subsequent exposition by Parks [102].

Theorem 12.3 *In Kaczmarz's algorithm, if $\mathbf{x}^{(0)} = \mathbf{0}$, then $\mathbf{x}^{(k)} \rightarrow \mathbf{x}^* = \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{b}$ as $k \rightarrow \infty$.* \square

Proof. We may assume without loss of generality that $\|\mathbf{a}_i\| = 1$, $i = 1, \dots, m$. For if not, we simply replace each \mathbf{a}_i by $\mathbf{a}_i/\|\mathbf{a}_i\|$ and each b_i by $b_i/\|\mathbf{a}_i\|$.

We first introduce the following notation. For each $j = 0, 1, 2, \dots$, let $R(j)$ denote the unique integer in $\{0, \dots, m-1\}$ satisfying $j = lm + R(j)$ for some integer l ; that is, $R(j)$ is the remainder that results if we divide j by m .

Using the notation above, we can write Kaczmarz's algorithm as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mu (b_{R(k)+1} - \mathbf{a}_{R(k)+1}^\top \mathbf{x}^{(k)}) \mathbf{a}_{R(k)+1}.$$

Using the identity $\|\mathbf{x} + \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + 2\langle \mathbf{x}, \mathbf{y} \rangle$, we obtain

$$\begin{aligned} \|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|^2 &= \|\mathbf{x}^{(k)} - \mathbf{x}^* + \mu (b_{R(k)+1} - \mathbf{a}_{R(k)+1}^\top \mathbf{x}^{(k)}) \mathbf{a}_{R(k)+1}\|^2 \\ &= \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2 + \mu^2 (b_{R(k)+1} - \mathbf{a}_{R(k)+1}^\top \mathbf{x}^{(k)})^2 \\ &\quad + 2\mu (b_{R(k)+1} - \mathbf{a}_{R(k)+1}^\top \mathbf{x}^{(k)}) \mathbf{a}_{R(k)+1}^\top (\mathbf{x}^{(k)} - \mathbf{x}^*). \end{aligned}$$

Substituting $\mathbf{a}_{R(k)+1}^\top \mathbf{x}^* = b_{R(k)+1}$ into this equation, we get

$$\begin{aligned} \|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|^2 &= \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2 - \mu(2 - \mu)(b_{R(k)+1} - \mathbf{a}_{R(k)+1}^\top \mathbf{x}^{(k)})^2 \\ &= \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2 - \mu(2 - \mu)(\mathbf{a}_{R(k)+1}^\top (\mathbf{x}^{(k)} - \mathbf{x}^*))^2. \end{aligned}$$

Because $0 < \mu < 2$, the second term on the right-hand side is nonnegative, and hence

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|^2 \leq \|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2.$$

Therefore, $\{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2\}$ is a nonincreasing sequence that is bounded below, because $\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2 \geq 0$ for all k . Hence, $\{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2\}$ converges (see Theorem 5.3). Furthermore, we may write

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2 = \|\mathbf{x}^{(0)} - \mathbf{x}^*\|^2 - \mu(2 - \mu) \sum_{i=0}^{k-1} (\mathbf{a}_{R(i)+1}^\top (\mathbf{x}^{(i)} - \mathbf{x}^*))^2.$$

Because $\{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2\}$ converges, we conclude that

$$\sum_{i=0}^{\infty} (\mathbf{a}_{R(i)+1}^\top (\mathbf{x}^{(i)} - \mathbf{x}^*))^2 < \infty,$$

which implies that

$$\mathbf{a}_{R(k)+1}^\top (\mathbf{x}^{(k)} - \mathbf{x}^*) \rightarrow 0.$$

Observe that

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|^2 = \mu^2 (\mathbf{b}_{R(k)+1} - \mathbf{a}_{R(k)+1}^\top \mathbf{x}^{(k)})^2 = \mu^2 (\mathbf{a}_{R(k)+1}^\top (\mathbf{x}^{(k)} - \mathbf{x}^*))^2$$

and therefore $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|^2 \rightarrow 0$. Note also that because $\{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2\}$ converges, $\{\mathbf{x}^{(k)}\}$ is a bounded sequence (see Theorem 5.2).

Following Kaczmarz [70], we introduce the notation $\mathbf{x}^{(r,s)} \triangleq \mathbf{x}^{(rm+s)}$, $r = 0, 1, 2, \dots$, $s = 0, \dots, m-1$. With this notation, we have, for each $s = 0, \dots, m-1$,

$$\mathbf{a}_{s+1}^\top (\mathbf{x}^{(r,s)} - \mathbf{x}^*) \rightarrow 0$$

as $r \rightarrow \infty$. Consider now the sequence $\{\mathbf{x}^{(r,0)} : r \geq 0\}$. Because this sequence is bounded, we conclude that it has a convergent subsequence—this follows from the Bolzano-Weierstrass theorem (see [2, p. 70]; see also Section 5.1 for a discussion of sequences and subsequences). Denote this convergent subsequence by $\{\mathbf{x}^{(r,0)} : r \in \mathcal{E}\}$, where \mathcal{E} is a subset of $\{0, 1, \dots\}$. Let \mathbf{z}^* be the limit of $\{\mathbf{x}^{(r,0)} : r \in \mathcal{E}\}$. Hence,

$$\mathbf{a}_1^\top (\mathbf{z}^* - \mathbf{x}^*) = 0.$$

Next, note that because $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|^2 \rightarrow 0$ as $k \rightarrow \infty$, we also have $\|\mathbf{x}^{(r,1)} - \mathbf{x}^{(r,0)}\|^2 \rightarrow 0$ as $r \rightarrow \infty$. Therefore, the subsequence $\{\mathbf{x}^{(r,1)} : r \in \mathcal{E}\}$ also converges to \mathbf{z}^* . Hence,

$$\mathbf{a}_2^\top (\mathbf{z}^* - \mathbf{x}^*) = 0.$$

Repeating the argument, we conclude that for each $i = 1, \dots, m$,

$$\mathbf{a}_i^\top (\mathbf{z}^* - \mathbf{x}^*) = 0.$$

In matrix notation, the equations above take the form

$$\mathbf{A}(\mathbf{z}^* - \mathbf{x}^*) = \mathbf{0}.$$

Now, $\mathbf{x}^{(k)} \in \mathcal{R}(\mathbf{A}^\top)$ for all k because $\mathbf{x}^{(0)} = \mathbf{0}$ (see Exercise 12.25). Therefore, $\mathbf{z}^* \in \mathcal{R}(\mathbf{A}^\top)$, because $\mathcal{R}(\mathbf{A}^\top)$ is closed. Hence, there exists \mathbf{y}^* such that $\mathbf{z}^* = \mathbf{A}^\top \mathbf{y}^*$. Thus,

$$\begin{aligned}\mathbf{A}(\mathbf{z}^* - \mathbf{x}^*) &= \mathbf{A}(\mathbf{A}^\top \mathbf{y}^* - \mathbf{A}^\top (\mathbf{A} \mathbf{A}^\top)^{-1} \mathbf{b}) \\ &= (\mathbf{A} \mathbf{A}^\top) \mathbf{y}^* - \mathbf{b} \\ &= \mathbf{0}.\end{aligned}$$

Because $\text{rank } \mathbf{A} = m$, $\mathbf{y}^* = (\mathbf{A} \mathbf{A}^\top)^{-1} \mathbf{b}$ and hence $\mathbf{z}^* = \mathbf{x}^*$. Therefore, the subsequence $\{\|\mathbf{x}^{r,0} - \mathbf{x}^*\|^2 : r \in \mathcal{E}\}$ converges to 0. Because $\{\|\mathbf{x}^{r,0} - \mathbf{x}^*\|^2 : r \in \mathcal{E}\}$ is a subsequence of the convergent sequence $\{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2\}$, we conclude that the sequence $\{\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2\}$ converges to 0; that is, $\mathbf{x}^{(k)} \rightarrow \mathbf{x}^*$. ■

For the case where $\mathbf{x}^{(0)} \neq \mathbf{0}$, Kaczmarz's algorithm converges to the unique point on $\{\mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}\}$ minimizing the distance $\|\mathbf{x} - \mathbf{x}^{(0)}\|$ (see Exercise 12.26).

If we set $\mu = 1$, Kaczmarz's algorithm has the property that at each iteration k , the “error” $b_{R(k)+1} - \mathbf{a}_{R(k)+1}^\top \mathbf{x}^{(k+1)}$ satisfies

$$b_{R(k)+1} - \mathbf{a}_{R(k)+1}^\top \mathbf{x}^{(k+1)} = 0$$

(see Exercise 12.28). Substituting $b_{R(k)+1} = \mathbf{a}_{R(k)+1}^\top \mathbf{x}^*$, we may write

$$\mathbf{a}_{R(k)+1}^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^*) = 0.$$

Hence, the difference between $\mathbf{x}^{(k+1)}$ and the solution \mathbf{x}^* is orthogonal to $\mathbf{a}_{R(k)+1}$. This property is illustrated in the following example.

Example 12.8 Let

$$\mathbf{A} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}.$$

In this case, $\mathbf{x}^* = [5, 3]^\top$. Figure 12.4 shows a few iterations of Kaczmarz's algorithm with $\mu = 1$ and $\mathbf{x}^{(0)} = \mathbf{0}$. We have $\mathbf{a}_1^\top = [1, -1]$, $\mathbf{a}_2^\top = [0, 1]$, $b_1 = 2$, $b_2 = 3$. In Figure 12.4, the diagonal line passing through the point $[2, 0]^\top$ corresponds to the set $\{\mathbf{x} : \mathbf{a}_1^\top \mathbf{x} = b_1\}$, and the horizontal line passing through the point $[0, 3]^\top$ corresponds to the set $\{\mathbf{x} : \mathbf{a}_2^\top \mathbf{x} = b_2\}$. To illustrate

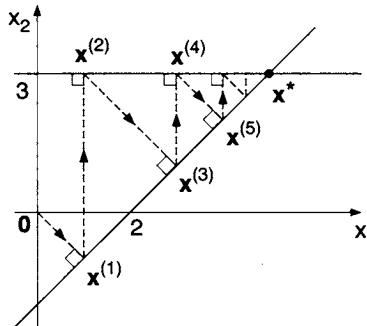


Figure 12.4 Iterations of Kaczmarz's algorithm in Example 12.8.

the algorithm, we perform three iterations:

$$\begin{aligned} \mathbf{x}^{(1)} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + (2 - 0) \frac{1}{2} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \\ \mathbf{x}^{(2)} &= \begin{bmatrix} 1 \\ -1 \end{bmatrix} + (3 - (-1)) \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \\ \mathbf{x}^{(3)} &= \begin{bmatrix} 1 \\ 3 \end{bmatrix} + (2 - (-2)) \frac{1}{2} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}. \end{aligned}$$

As Figure 12.4 illustrates, the property

$$\mathbf{a}_{R(k)+1}^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^*) = 0$$

holds at every iteration. Note the convergence of the iterations of the algorithm to the solution. ■

12.5 Solving Linear Equations in General

Consider the general problem of solving a system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$, and $\text{rank } \mathbf{A} = r$. Note that we always have $r \leq \min\{m, n\}$. In the case where $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\text{rank } \mathbf{A} = n$, the unique solution to the equation above has the form $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$. Thus, to solve the problem in this case it is enough to know the inverse \mathbf{A}^{-1} . In this section we analyze a general approach to solving $\mathbf{A}\mathbf{x} = \mathbf{b}$. The approach involves defining a *pseudoinverse* or *generalized inverse* of a given matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, which plays the role of

\mathbf{A}^{-1} when \mathbf{A} does not have an inverse (e.g., when \mathbf{A} is not a square matrix). In particular, we discuss the *Moore-Penrose inverse* of a given matrix \mathbf{A} , denoted \mathbf{A}^\dagger .

In our discussion of the Moore-Penrose inverse we use the fact that a nonzero matrix of rank r can be expressed as the product of a matrix of full column rank r and a matrix of full row rank r . Such a factorization is referred to as a *full-rank factorization*, a term which in this context was proposed by Gantmacher [45] and Ben-Israel and Greville [6]. We state and prove the above result in the following lemma.

Lemma 12.3 Full-Rank Factorization. *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\text{rank } \mathbf{A} = r \leq \min\{m, n\}$. Then, there exist matrices $\mathbf{B} \in \mathbb{R}^{m \times r}$ and $\mathbf{C} \in \mathbb{R}^{r \times n}$ such that*

$$\mathbf{A} = \mathbf{BC},$$

where

$$\text{rank } \mathbf{A} = \text{rank } \mathbf{B} = \text{rank } \mathbf{C} = r.$$

□

Proof. Because $\text{rank } \mathbf{A} = r$, we can find r linearly independent columns of \mathbf{A} . Without loss of generality, let $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_r$ be such columns, where \mathbf{a}_i is the i th column of \mathbf{A} . The remaining columns of \mathbf{A} can be expressed as linear combinations of $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_r$. Thus, a possible choice for the full-rank matrices \mathbf{B} and \mathbf{C} are

$$\mathbf{B} = [\mathbf{a}_1, \dots, \mathbf{a}_r] \in \mathbb{R}^{m \times r},$$

$$\mathbf{C} = \begin{bmatrix} 1 & \cdots & 0 & c_{1,r+1} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & c_{r,r+1} & \cdots & c_{r,n} \end{bmatrix} \in \mathbb{R}^{r \times n},$$

where the entries $c_{i,j}$ are such that for each $j = r + 1, \dots, n$, we have $\mathbf{a}_j = c_{1,j}\mathbf{a}_1 + \cdots + c_{r,j}\mathbf{a}_r$. Thus, $\mathbf{A} = \mathbf{BC}$. ■

Note that if $m < n$ and $\text{rank } \mathbf{A} = m$, then we can take

$$\mathbf{B} = \mathbf{I}_m, \quad \mathbf{C} = \mathbf{A},$$

where \mathbf{I}_m is the $m \times m$ identity matrix. If, on the other hand, $m > n$ and $\text{rank } \mathbf{A} = n$, then we can take

$$\mathbf{B} = \mathbf{A}, \quad \mathbf{C} = \mathbf{I}_n.$$

Example 12.9 Let \mathbf{A} be given by

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & -2 & 5 \\ 1 & 0 & -3 & 2 \\ 3 & -1 & -13 & 5 \end{bmatrix}.$$

Note that $\text{rank } \mathbf{A} = 2$. We can write a full-rank factorization of \mathbf{A} based on the proof of Lemma 12.3:

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \\ 3 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 1 & 4 & 1 \end{bmatrix} = \mathbf{BC}.$$

■

We now introduce the *Moore-Penrose inverse* and discuss its existence and uniqueness. For this, we first consider the matrix equation

$$\mathbf{AXA} = \mathbf{A},$$

where $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a given matrix and $\mathbf{X} \in \mathbb{R}^{n \times m}$ is a matrix we wish to determine. Observe that if \mathbf{A} is a nonsingular square matrix, then the equation above has the unique solution $\mathbf{X} = \mathbf{A}^{-1}$. We now define the Moore-Penrose inverse, also called the pseudoinverse or generalized inverse.

Definition 12.1 Given $\mathbf{A} \in \mathbb{R}^{m \times n}$, a matrix $\mathbf{A}^\dagger \in \mathbb{R}^{n \times m}$ is called a *pseudoinverse* of the matrix \mathbf{A} if

$$\mathbf{AA}^\dagger \mathbf{A} = \mathbf{A},$$

and there exist matrices $\mathbf{U} \in \mathbb{R}^{n \times n}$, $\mathbf{V} \in \mathbb{R}^{m \times m}$ such that

$$\mathbf{A}^\dagger = \mathbf{U}\mathbf{A}^\top \quad \text{and} \quad \mathbf{A}^\dagger = \mathbf{A}^\top\mathbf{V}.$$

■

The requirement $\mathbf{A}^\dagger = \mathbf{U}\mathbf{A}^\top = \mathbf{A}^\top\mathbf{V}$ can be interpreted as follows. Each row of the pseudoinverse matrix \mathbf{A}^\dagger of \mathbf{A} is a linear combination of the rows of \mathbf{A}^\top , and each column of \mathbf{A}^\dagger is a linear combination of the columns of \mathbf{A}^\top .

For the case in which a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m \geq n$ and $\text{rank } \mathbf{A} = n$, we can easily check that the following is a pseudoinverse of \mathbf{A} :

$$\mathbf{A}^\dagger = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top.$$

Indeed, $\mathbf{A}(\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\mathbf{A} = \mathbf{A}$, and if we define $\mathbf{U} = (\mathbf{A}^\top\mathbf{A})^{-1}$ and $\mathbf{V} = \mathbf{A}(\mathbf{A}^\top\mathbf{A})^{-1}(\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top$, then $\mathbf{A}^\dagger = \mathbf{U}\mathbf{A}^\top = \mathbf{A}^\top\mathbf{V}$. Note that, in fact, we have $\mathbf{A}^\dagger\mathbf{A} = \mathbf{I}_n$. For this reason, $(\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top$ is often called the *left pseudoinverse* of \mathbf{A} . This formula also appears in least-squares analysis (see Section 12.1).

For the case in which a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m \leq n$ and $\text{rank } \mathbf{A} = m$, we can easily check, as we did in the previous case, that the following is a pseudoinverse of \mathbf{A} :

$$\mathbf{A}^\dagger = \mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}.$$

Note that in this case we have $\mathbf{A}\mathbf{A}^\dagger = \mathbf{I}_m$. For this reason, $\mathbf{A}^\top(\mathbf{A}\mathbf{A}^\top)^{-1}$ is often called the *right pseudoinverse* of \mathbf{A} . This formula also appears in the problem of minimizing $\|\mathbf{x}\|$ subject to $\mathbf{Ax} = \mathbf{b}$ (see Section 12.3).

Theorem 12.4 *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$. If a pseudoinverse \mathbf{A}^\dagger of \mathbf{A} exists, then it is unique.* \square

Proof. Let \mathbf{A}_1^\dagger and \mathbf{A}_2^\dagger be pseudoinverses of \mathbf{A} . We show that $\mathbf{A}_1^\dagger = \mathbf{A}_2^\dagger$. By definition,

$$\mathbf{A}\mathbf{A}_1^\dagger\mathbf{A} = \mathbf{A}\mathbf{A}_2^\dagger\mathbf{A} = \mathbf{A},$$

and there are matrices $\mathbf{U}_1, \mathbf{U}_2 \in \mathbb{R}^{n \times n}$, $\mathbf{V}_1, \mathbf{V}_2 \in \mathbb{R}^{m \times m}$ such that

$$\begin{aligned}\mathbf{A}_1^\dagger &= \mathbf{U}_1\mathbf{A}^\top = \mathbf{A}^\top\mathbf{V}_1, \\ \mathbf{A}_2^\dagger &= \mathbf{U}_2\mathbf{A}^\top = \mathbf{A}^\top\mathbf{V}_2.\end{aligned}$$

Let

$$\mathbf{D} = \mathbf{A}_2^\dagger - \mathbf{A}_1^\dagger, \mathbf{U} = \mathbf{U}_2 - \mathbf{U}_1, \mathbf{V} = \mathbf{V}_2 - \mathbf{V}_1.$$

Then, we have

$$\mathbf{O} = \mathbf{A}\mathbf{D}\mathbf{A}, \mathbf{D} = \mathbf{U}\mathbf{A}^\top = \mathbf{A}^\top\mathbf{V}.$$

Therefore, using the two equations above, we have

$$(\mathbf{D}\mathbf{A})^\top\mathbf{D}\mathbf{A} = \mathbf{A}^\top\mathbf{D}^\top\mathbf{D}\mathbf{A} = \mathbf{A}^\top\mathbf{V}^\top\mathbf{A}\mathbf{D}\mathbf{A} = \mathbf{O},$$

which implies that

$$\mathbf{D}\mathbf{A} = \mathbf{O}.$$

On the other hand, because $\mathbf{D}\mathbf{A} = \mathbf{O}$, we have

$$\mathbf{D}\mathbf{D}^\top = \mathbf{D}\mathbf{A}\mathbf{U}^\top = \mathbf{O},$$

which implies that

$$\mathbf{D} = \mathbf{A}_2^\dagger - \mathbf{A}_1^\dagger = \mathbf{O}$$

and hence

$$\mathbf{A}_2^\dagger = \mathbf{A}_1^\dagger.$$

■

From Theorem 12.4, we know that if a pseudoinverse matrix exists, then it is unique. Our goal now is to show that the pseudoinverse always exists. In fact, we show that the pseudoinverse of any given matrix \mathbf{A} is given by the formula

$$\mathbf{A}^\dagger = \mathbf{C}^\dagger\mathbf{B}^\dagger,$$

where \mathbf{B}^\dagger and \mathbf{C}^\dagger are the pseudoinverses of the matrices \mathbf{B} and \mathbf{C} that form a full-rank factorization of \mathbf{A} ; that is, $\mathbf{A} = \mathbf{BC}$, where \mathbf{B} and \mathbf{C} are of full

rank (see Lemma 12.3). Note that we already know how to compute \mathbf{B}^\dagger and \mathbf{C}^\dagger :

$$\mathbf{B}^\dagger = (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top, \quad \mathbf{C}^\dagger = \mathbf{C}^\top (\mathbf{C} \mathbf{C}^\top)^{-1}.$$

Theorem 12.5 *Let a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ have a full-rank factorization $\mathbf{A} = \mathbf{BC}$, with $\text{rank } \mathbf{A} = \text{rank } \mathbf{B} = \text{rank } \mathbf{C} = r$, $\mathbf{B} \in \mathbb{R}^{m \times r}$, $\mathbf{C} \in \mathbb{R}^{r \times n}$. Then,*

$$\mathbf{A}^\dagger = \mathbf{C}^\dagger \mathbf{B}^\dagger.$$

□

Proof. We show that

$$\mathbf{A}^\dagger = \mathbf{C}^\dagger \mathbf{B}^\dagger = \mathbf{C}^\top (\mathbf{C} \mathbf{C}^\top)^{-1} (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top$$

satisfies the conditions of Definition 12.1 for a pseudoinverse. Indeed, first observe that

$$\mathbf{AC}^\dagger \mathbf{B}^\dagger \mathbf{A} = \mathbf{BCC}^\top (\mathbf{C} \mathbf{C}^\top)^{-1} (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{BC} = \mathbf{BC} = \mathbf{A}.$$

Next, define

$$\mathbf{U} = \mathbf{C}^\top (\mathbf{C} \mathbf{C}^\top)^{-1} (\mathbf{B}^\top \mathbf{B})^{-1} (\mathbf{C} \mathbf{C}^\top)^{-1} \mathbf{C}$$

and

$$\mathbf{V} = \mathbf{B} (\mathbf{B}^\top \mathbf{B})^{-1} (\mathbf{C} \mathbf{C}^\top)^{-1} (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top.$$

It is easy to verify that the matrices \mathbf{U} and \mathbf{V} above satisfy

$$\mathbf{A}^\dagger = \mathbf{C}^\dagger \mathbf{B}^\dagger = \mathbf{U} \mathbf{A}^\top = \mathbf{A}^\top \mathbf{V}.$$

Hence,

$$\mathbf{A}^\dagger = \mathbf{C}^\dagger \mathbf{B}^\dagger$$

is the pseudoinverse of \mathbf{A} . ■

Example 12.10 *Continued from Example 12.9.* Recall that

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & -2 & 5 \\ 1 & 0 & -3 & 2 \\ 3 & -1 & -13 & 5 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \\ 3 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 1 & 4 & 1 \end{bmatrix} = \mathbf{BC}.$$

We compute

$$\mathbf{B}^\dagger = (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top = \frac{1}{27} \begin{bmatrix} 5 & 2 & 5 \\ 16 & 1 & -11 \end{bmatrix}$$

and

$$\mathbf{C}^\dagger = \mathbf{C}^\top (\mathbf{C} \mathbf{C}^\top)^{-1} = \frac{1}{76} \begin{bmatrix} 9 & 5 \\ 5 & 7 \\ -7 & 13 \\ 23 & 17 \end{bmatrix}.$$

Thus,

$$\mathbf{A}^\dagger = \mathbf{C}^\dagger \mathbf{B}^\dagger = \frac{1}{2052} \begin{bmatrix} 125 & 23 & -10 \\ 137 & 17 & -52 \\ 173 & -1 & -178 \\ 387 & 63 & -72 \end{bmatrix}.$$

■

We emphasize that the formula $\mathbf{A}^\dagger = \mathbf{C}^\dagger \mathbf{B}^\dagger$ does not necessarily hold if $\mathbf{A} = \mathbf{BC}$ is not a full-rank factorization. The following example, which is taken from [45], illustrates this point.

Example 12.11 Let

$$\mathbf{A} = \begin{bmatrix} 1 \end{bmatrix}.$$

Obviously, $\mathbf{A}^\dagger = \mathbf{A}^{-1} = \mathbf{A} = [1]$. Observe that \mathbf{A} can be represented as

$$\mathbf{A} = [0 \ 1] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \mathbf{BC}.$$

The above is not a full-rank factorization of \mathbf{A} . Let us now compute \mathbf{B}^\dagger and \mathbf{C}^\dagger . We have

$$\mathbf{B}^\dagger = \mathbf{B}^\top (\mathbf{BB}^\top)^{-1} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

$$\mathbf{C}^\dagger = (\mathbf{C}^\top \mathbf{C})^{-1} \mathbf{C}^\top = \begin{bmatrix} 1/2 & 1/2 \end{bmatrix}.$$

(Note that the formulas for \mathbf{B}^\dagger and \mathbf{C}^\dagger here are different from those in Example 12.10 because of the dimensions of \mathbf{B} and \mathbf{C} in this example.) Thus,

$$\mathbf{C}^\dagger \mathbf{B}^\dagger = \begin{bmatrix} 1/2 \end{bmatrix},$$

which is not equal to \mathbf{A}^\dagger .

■

We can simplify the expression

$$\mathbf{A}^\dagger = \mathbf{C}^\dagger \mathbf{B}^\dagger = \mathbf{C}^\top (\mathbf{CC}^\top)^{-1} (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top$$

to

$$\mathbf{A}^\dagger = \mathbf{C}^\top (\mathbf{B}^\top \mathbf{AC}^\top)^{-1} \mathbf{B}^\top.$$

The expression above is easily verified simply by substituting $\mathbf{A} = \mathbf{BC}$. This explicit formula for \mathbf{A}^\dagger is attributed to C. C. MacDuffee by Ben-Israel and Greville [6]. Ben-Israel and Greville report that around 1959, MacDuffee was the first to point out that a full-rank factorization of \mathbf{A} leads to the above

explicit formula. However, they mention that MacDuffee did it in a private communication, so there is no published work by MacDuffee that contains the result.

We now prove two important properties of \mathbf{A}^\dagger in the context of solving a system of linear equations $\mathbf{Ax} = \mathbf{b}$.

Theorem 12.6 Consider a system of linear equations $\mathbf{Ax} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\text{rank } \mathbf{A} = r$. The vector $\mathbf{x}^* = \mathbf{A}^\dagger \mathbf{b}$ minimizes $\|\mathbf{Ax} - \mathbf{b}\|^2$ on \mathbb{R}^n . Furthermore, among all vectors in \mathbb{R}^n that minimize $\|\mathbf{Ax} - \mathbf{b}\|^2$, the vector $\mathbf{x}^* = \mathbf{A}^\dagger \mathbf{b}$ is the unique vector with minimal norm. \square

Proof. We first show that $\mathbf{x}^* = \mathbf{A}^\dagger \mathbf{b}$ minimizes $\|\mathbf{Ax} - \mathbf{b}\|^2$ over \mathbb{R}^n . To this end, observe that for any $\mathbf{x} \in \mathbb{R}^n$,

$$\begin{aligned}\|\mathbf{Ax} - \mathbf{b}\|^2 &= \|\mathbf{A}(\mathbf{x} - \mathbf{x}^*) + \mathbf{Ax}^* - \mathbf{b}\|^2 \\ &= \|\mathbf{A}(\mathbf{x} - \mathbf{x}^*)\|^2 + \|\mathbf{Ax}^* - \mathbf{b}\|^2 + 2[\mathbf{A}(\mathbf{x} - \mathbf{x}^*)]^\top (\mathbf{Ax}^* - \mathbf{b}).\end{aligned}$$

We now show that

$$[\mathbf{A}(\mathbf{x} - \mathbf{x}^*)]^\top (\mathbf{Ax}^* - \mathbf{b}) = 0.$$

Indeed,

$$\begin{aligned}[\mathbf{A}(\mathbf{x} - \mathbf{x}^*)]^\top (\mathbf{Ax}^* - \mathbf{b}) &= (\mathbf{x} - \mathbf{x}^*)^\top (\mathbf{A}^\top \mathbf{Ax}^* - \mathbf{A}^\top \mathbf{b}) \\ &= (\mathbf{x} - \mathbf{x}^*)^\top (\mathbf{A}^\top \mathbf{A} \mathbf{A}^\dagger \mathbf{b} - \mathbf{A}^\top \mathbf{b}).\end{aligned}$$

However,

$$\mathbf{A}^\top \mathbf{A} \mathbf{A}^\dagger = \mathbf{C}^\top \mathbf{B}^\top \mathbf{B} \mathbf{C} \mathbf{C}^\top (\mathbf{C} \mathbf{C}^\top)^{-1} (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top = \mathbf{A}^\top.$$

Hence,

$$[\mathbf{A}(\mathbf{x} - \mathbf{x}^*)]^\top (\mathbf{Ax}^* - \mathbf{b}) = (\mathbf{x} - \mathbf{x}^*)^\top (\mathbf{A}^\top \mathbf{b} - \mathbf{A}^\top \mathbf{b}) = 0.$$

Thus, we have

$$\|\mathbf{Ax} - \mathbf{b}\|^2 = \|\mathbf{A}(\mathbf{x} - \mathbf{x}^*)\|^2 + \|\mathbf{Ax}^* - \mathbf{b}\|^2.$$

Because

$$\|\mathbf{A}(\mathbf{x} - \mathbf{x}^*)\|^2 \geq 0,$$

we obtain

$$\|\mathbf{Ax} - \mathbf{b}\|^2 \geq \|\mathbf{Ax}^* - \mathbf{b}\|^2$$

and thus \mathbf{x}^* minimizes $\|\mathbf{Ax} - \mathbf{b}\|^2$.

We now show that among all \mathbf{x} that minimize $\|\mathbf{Ax} - \mathbf{b}\|^2$, the vector $\mathbf{x}^* = \mathbf{A}^\dagger \mathbf{b}$ is the unique vector with minimum norm. So let $\tilde{\mathbf{x}}$ be a vector minimizing $\|\mathbf{Ax} - \mathbf{b}\|^2$. We have

$$\begin{aligned}\|\tilde{\mathbf{x}}\|^2 &= \|(\tilde{\mathbf{x}} - \mathbf{x}^*) + \mathbf{x}^*\|^2 \\ &= \|\tilde{\mathbf{x}} - \mathbf{x}^*\|^2 + \|\mathbf{x}^*\|^2 + 2\mathbf{x}^{*\top}(\tilde{\mathbf{x}} - \mathbf{x}^*).\end{aligned}$$

Observe that

$$\mathbf{x}^{*\top}(\tilde{\mathbf{x}} - \mathbf{x}^*) = 0.$$

To see this, note that

$$\begin{aligned}\mathbf{x}^{*\top}(\tilde{\mathbf{x}} - \mathbf{x}^*) &= (\mathbf{A}^\dagger \mathbf{b})^\top(\tilde{\mathbf{x}} - \mathbf{A}^\dagger \mathbf{b}) \\ &= \mathbf{b}^\top \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-\top}(\mathbf{C} \mathbf{C}^\top)^{-\top} \mathbf{C}(\tilde{\mathbf{x}} - \mathbf{C}^\top(\mathbf{C} \mathbf{C}^\top)^{-1}(\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{b}) \\ &= \mathbf{b}^\top \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-\top}(\mathbf{C} \mathbf{C}^\top)^{-\top}[\mathbf{C} \tilde{\mathbf{x}} - (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{b}],\end{aligned}$$

where the superscript $-\top$ denotes the transpose of the inverse. Now, $\|\mathbf{Ax} - \mathbf{b}\|^2 = \|\mathbf{B}(\mathbf{Cx}) - \mathbf{b}\|^2$. Because $\tilde{\mathbf{x}}$ minimizes $\|\mathbf{Ax} - \mathbf{b}\|^2$ and \mathbf{C} is of full rank, then $\mathbf{y}^* = \mathbf{C}\tilde{\mathbf{x}}$ minimizes $\|\mathbf{By} - \mathbf{b}\|^2$ over \mathbb{R}^r (see Exercise 12.29). Because \mathbf{B} is of full rank, by Theorem 12.1, we have $\mathbf{C}\tilde{\mathbf{x}} = \mathbf{y}^* = (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{b}$. Substituting this into the equation above, we get $\mathbf{x}^{*\top}(\tilde{\mathbf{x}} - \mathbf{x}^*) = 0$.

Therefore, we have

$$\|\tilde{\mathbf{x}}\|^2 = \|\mathbf{x}^*\|^2 + \|\tilde{\mathbf{x}} - \mathbf{x}^*\|^2.$$

For all $\tilde{\mathbf{x}} \neq \mathbf{x}^*$, we have

$$\|\tilde{\mathbf{x}} - \mathbf{x}^*\|^2 > 0,$$

and hence

$$\|\tilde{\mathbf{x}}\|^2 > \|\mathbf{x}^*\|^2$$

or, equivalently,

$$\|\tilde{\mathbf{x}}\| > \|\mathbf{x}^*\|.$$

Hence, among all vectors minimizing $\|\mathbf{Ax} - \mathbf{b}\|^2$, the vector $\mathbf{x}^* = \mathbf{A}^\dagger \mathbf{b}$ is the unique vector with minimum norm. ■

The generalized inverse has the following useful properties (see Exercise 12.30):

- a. $(\mathbf{A}^\top)^\dagger = (\mathbf{A}^\dagger)^\top$.
- b. $(\mathbf{A}^\dagger)^\dagger = \mathbf{A}$.

These two properties are similar to those that are satisfied by the usual matrix inverse. However, we point out that the property $(\mathbf{A}_1 \mathbf{A}_2)^\dagger = \mathbf{A}_2^\dagger \mathbf{A}_1^\dagger$ does not hold in general (see Exercise 12.32).

Finally, it is important to note that we can define the generalized inverse in an alternative way, following the definition of Penrose. Specifically, the Penrose definition of the generalized inverse of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the unique matrix $\mathbf{A}^\dagger \in \mathbb{R}^{n \times m}$ satisfying the following properties:

1. $\mathbf{A}\mathbf{A}^\dagger \mathbf{A} = \mathbf{A}$.
2. $\mathbf{A}^\dagger \mathbf{A}\mathbf{A}^\dagger = \mathbf{A}^\dagger$.

$$3. (\mathbf{A}\mathbf{A}^\dagger)^\top = \mathbf{A}\mathbf{A}^\dagger.$$

$$4. (\mathbf{A}^\dagger\mathbf{A})^\top = \mathbf{A}^\dagger\mathbf{A}.$$

The Penrose definition above is equivalent to Definition 12.1 (see Exercise 12.31). For more information on generalized inverses and their applications, we refer the reader to the books by Ben-Israel and Greville [6] and Campbell and Meyer [23].

EXERCISES

12.1 A rock is accelerated to 3, 5, and 6 m/s² by applying forces of 1, 2, and 3 N, respectively. Assuming that Newton's law $F = ma$ holds, where F is the force and a is the acceleration, estimate the mass m of the rock using the least-squares method.

12.2 A spring is stretched to lengths $L = 3, 4$, and 5 cm under applied forces $F = 1, 2$, and 4 N, respectively. Assuming that Hooke's law $L = a + bF$ holds, estimate the normal length a and spring constant b using the least-squares approach.

12.3 Suppose that we perform an experiment to calculate the gravitational constant g as follows. We drop a ball from a certain height and measure its distance from the original point at certain time instants. The results of the experiment are shown in the following table.

Time (seconds)	1.00	2.00	3.00
Distance (meters)	5.00	19.5	44.0

The equation relating the distance s and the time t at which s is measured is given by

$$s = \frac{1}{2}gt^2.$$

- a. Find a least-squares estimate of g using the experimental results from the table above.
- b. Suppose that we take an additional measurement at time 4.00 and obtain a distance of 78.5. Use the recursive least-squares algorithm to calculate an updated least-squares estimate of g .

12.4 Suppose that we have a speech signal, represented as a finite sequence of real numbers x_1, x_2, \dots, x_n . Suppose that we record this signal onto magnetic tape. The recorded speech signal is represented by another sequence of real numbers y_1, y_2, \dots, y_n .

Suppose that we model the recording process as a simple scaling of the original signal (i.e., we believe that a good model of the relationship between the recorded signal and the original signal is $y_i = \alpha x_i$ for some constant α that does not depend on i). Suppose that we know exactly the original signal x_1, x_2, \dots, x_n as well as the recorded signal y_1, y_2, \dots, y_n . Use the least-squares method to find a formula for estimating the scale factor α given this data. (You may assume that at least one x_i is nonzero.)

12.5 Suppose that we wish to estimate the value of the resistance R of a resistor. Ohm's law states that if V is the voltage across the resistor and I is the current through the resistor, then $V = IR$. To estimate R , we apply a 1-ampere current through the resistor and measure the voltage across it. We perform the experiment on n voltage-measuring devices and obtain measurements of V_1, \dots, V_n . Show that the least-squares estimate of R is simply the average of V_1, \dots, V_n .

12.6 The table below shows the stock prices for three companies, X, Y, and Z, tabulated over three days:

	Day		
	1	2	3
X	6	4	5
Y	1	1	3
Z	2	1	2

Suppose that an investment analyst proposes a model for the predicting the stock price of X based on those of Y and Z:

$$p_X = ap_Y + bp_Z,$$

where p_X , p_Y , and p_Z are the stock prices of X, Y, and Z, respectively, and a and b are real-valued parameters. Calculate the least-squares estimate of parameters a and b based on the data in the table above.

12.7 We are given two mixtures, A and B. Mixture A contains 30% gold, 40% silver, and 30% platinum, whereas mixture B contains 10% gold, 20% silver, and 70% platinum (all percentages of weight). We wish to determine the ratio of the weight of mixture A to the weight of mixture B such that we have as close as possible to a total of 5 ounces of gold, 3 ounces of silver, and 4 ounces of platinum. Formulate and solve the problem using the linear least-squares method.

12.8 *Background:* If $\mathbf{A}\mathbf{x} + \mathbf{w} = \mathbf{b}$, where \mathbf{w} is a “white noise” vector, then define the *least-squares estimate* of \mathbf{x} given \mathbf{b} to be the solution to the problem

$$\text{minimize } \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2.$$

This problem is related to Wiener filtering.

Application: Suppose that a given speech signal $\{u_k : k = 1, \dots, n\}$ (with $u_k \in \mathbb{R}$) is transmitted over a telephone cable with input-output behavior given by $y_k = ay_{k-1} + bu_k + v_k$, where, at each time k , $y_k \in \mathbb{R}$ is the output, $u_k \in \mathbb{R}$ is the input (speech signal value), and v_k represents white noise. The parameters a and b are fixed known constants, and the initial condition is $y_0 = 0$.

We can measure the signal $\{y_k\}$ at the output of the telephone cable, but we cannot directly measure the desired signal $\{u_k\}$ or the noise signal $\{v_k\}$. Derive a formula for the linear least-squares estimate of the signal $\{u_k : k = 1, \dots, n\}$ given the signal $\{y_k : k = 1, \dots, n\}$.

Note: Even though the vector $\mathbf{v} = [v_1, \dots, v_n]^\top$ is a white noise vector, the vector $\mathbf{D}\mathbf{v}$ (where \mathbf{D} is a matrix) is, in general, not.

12.9 Line Fitting. Let $[x_1, y_1]^\top, \dots, [x_p, y_p]^\top$, $p \geq 2$, be points in \mathbb{R}^2 . We wish to find the straight line of best fit through these points (“best” in the sense that the total squared error is minimized); that is, we wish to find $a^*, b^* \in \mathbb{R}$ to minimize

$$f(a, b) = \sum_{i=1}^p (ax_i + b - y_i)^2.$$

Assume that the x_i , $i = 1, \dots, p$, are not all equal. Show that there exist unique parameters a^* and b^* for the line of best fit, and find the parameters in terms of the following quantities:

$$\begin{aligned}\overline{X} &= \frac{1}{p} \sum_{i=1}^p x_i, \\ \overline{Y} &= \frac{1}{p} \sum_{i=1}^p y_i, \\ \overline{X^2} &= \frac{1}{p} \sum_{i=1}^p x_i^2, \\ \overline{Y^2} &= \frac{1}{p} \sum_{i=1}^p y_i^2, \\ \overline{XY} &= \frac{1}{p} \sum_{i=1}^p x_i y_i.\end{aligned}$$

12.10 Suppose that we take measurements of a sinusoidal signal $y(t) = \sin(\omega t + \theta)$ at times t_1, \dots, t_p , and obtain values y_1, \dots, y_p , where $-\pi/2 \leq \omega t_i + \theta \leq \pi/2$, $i = 1, \dots, p$, and the t_i are not all equal. We wish to determine the values of the frequency ω and phase θ .

- a. Express the problem as a system of linear equations.
- b. Find the least-squares estimate of ω and θ based on part a. Use the following notation:

$$\begin{aligned}\bar{T} &= \frac{1}{p} \sum_{i=1}^p t_i, \\ \bar{T^2} &= \frac{1}{p} \sum_{i=1}^p t_i^2, \\ \bar{TY} &= \frac{1}{p} \sum_{i=1}^p t_i \arcsin y_i, \\ \bar{Y} &= \frac{1}{p} \sum_{i=1}^p \arcsin y_i.\end{aligned}$$

12.11 We are given a point $[x_0, y_0]^\top \in \mathbb{R}^2$. Consider the straight line on the \mathbb{R}^2 plane given by the equation $y = mx$. Using a least-squares formulation, find the point on the straight line that is closest to the given point $[x_0, y_0]$, where the measure of closeness is in terms of the Euclidean norm on \mathbb{R}^2 .

Hint: The given line can be expressed as the range of the matrix $\mathbf{A} = [1, m]^\top$.

12.12 Consider the affine function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ of the form $f(\mathbf{x}) = \mathbf{a}^\top \mathbf{x} + c$, where $\mathbf{a} \in \mathbb{R}^n$ and $c \in \mathbb{R}$.

- a. We are given a set of p pairs $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_p, y_p)$, where $\mathbf{x}_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$, $i = 1, \dots, p$. We wish to find the affine function of best fit to these points, where “best” is in the sense of minimizing the total square error

$$\sum_{i=1}^p (f(\mathbf{x}_i) - y_i)^2.$$

Formulate the above as an optimization problem of the form: minimize $\|\mathbf{Az} - \mathbf{b}\|^2$ with respect to \mathbf{z} . Specify the dimensions of \mathbf{A} , \mathbf{z} , and \mathbf{b} .

- b. Suppose that the points satisfy

$$\mathbf{x}_1 + \cdots + \mathbf{x}_p = \mathbf{0}$$

and

$$y_1 \mathbf{x}_1 + \cdots + y_p \mathbf{x}_p = \mathbf{0}.$$

Find the affine function of best fit in this case, assuming that it exists and is unique.



Figure 12.5 Input-output system in Exercise 12.13.

12.13 For the system shown in Figure 12.5, consider a set of input-output pairs $(u_1, y_1), \dots, (u_n, y_n)$, where $u_k \in \mathbb{R}$, $y_k \in \mathbb{R}$, $k = 1, \dots, n$.

- Suppose that we wish to find the best linear estimate of the system based on the input-output data above. In other words, we wish to find a $\hat{\theta}_n \in \mathbb{R}$ to fit the model $y_k = \hat{\theta}_n u_k$, $k = 1, \dots, n$. Using the least-squares approach, derive a formula for $\hat{\theta}_n$ based on u_1, \dots, u_n and y_1, \dots, y_n .
- Suppose that the data in part a are generated by

$$y_k = \theta u_k + e_k,$$

where $\theta \in \mathbb{R}$ and $u_k = 1$ for all k . Show that the parameter $\hat{\theta}_n$ in part a converges to θ as $n \rightarrow \infty$ if and only if

$$\lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n e_k = 0.$$

12.14 Consider a discrete-time linear system $x_{k+1} = ax_k + bu_k$, where u_k is the input at time k , x_k is the output at time k , and $a, b \in \mathbb{R}$ are system parameters. Suppose that we apply a constant input $u_k = 1$ for all $k \geq 0$ and measure the first four values of the output to be $x_0 = 0$, $x_1 = 1$, $x_2 = 2$, $x_3 = 8$. Find the least-squares estimate of a and b based on the data above.

12.15 Consider a discrete-time linear system $x_{k+1} = ax_k + bu_k$, where u_k is the input at time k , x_k is the output at time k , and $a, b \in \mathbb{R}$ are system parameters. Given the first $n + 1$ values of the impulse response h_0, \dots, h_n , find the least-squares estimate of a and b . You may assume that at least one h_k is nonzero.

Note: The *impulse response* is the output sequence resulting from an input of $u_0 = 1$, $u_k = 0$ for $k \neq 0$ and zero initial condition $x_0 = 0$.

12.16 Consider a discrete-time linear system $x_{k+1} = ax_k + bu_k$, where u_k is the input at time k , x_k is the output at time k , and $a, b \in \mathbb{R}$ are system parameters. Given the first $n + 1$ values of the step response s_0, \dots, s_n , where $n > 1$, find the least-squares estimate of a and b . You may assume that at least one s_k is nonzero.

Note: The *step response* is the output sequence resulting from an input of $u_k = 1$ for $k \geq 0$, and zero initial condition $x_0 = 0$ (i.e., $s_0 = x_0 = 0$).

12.17 Consider a known discrete-time signal on the time interval $\{1, \dots, n\}$, represented by the vector $\mathbf{x} \in \mathbb{R}^n$ (x_i is the value of the signal at time i). We transmit the signal $a\mathbf{x}$ over a communication channel, where $a \in \mathbb{R}$ represents the “amplitude” of the transmission, a quantity unknown to the receiver. The receiver receives a signal $\mathbf{y} \in \mathbb{R}^n$, which is a distorted version of the transmitted signal (so that \mathbf{y} may not be equal to $a\mathbf{x}$ for any a). Formulate the problem of estimating the quantity a according to a least-squares criterion, and solve it (stating whatever appropriate assumptions are necessary, if any).

12.18 Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $m \geq n$, and $\text{rank } \mathbf{A} = n$. Consider the constrained optimization problem

$$\begin{aligned} & \text{minimize} \quad \frac{1}{2} \mathbf{x}^\top \mathbf{x} - \mathbf{x}^\top \mathbf{b} \\ & \text{subject to} \quad \mathbf{x} \in \mathcal{R}(\mathbf{A}), \end{aligned}$$

where $\mathcal{R}(\mathbf{A})$ denotes the range of \mathbf{A} . Derive an expression for the global minimizer of this problem in terms of \mathbf{A} and \mathbf{b} .

12.19 Solve the problem

$$\begin{aligned} & \text{minimize} \quad \|\mathbf{x} - \mathbf{x}_0\| \\ & \text{subject to} \quad \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \mathbf{x} = 1, \end{aligned}$$

where $\mathbf{x}_0 = [0, -3, 0]^\top$.

12.20 Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $m \leq n$, $\text{rank } \mathbf{A} = m$, and $\mathbf{x}_0 \in \mathbb{R}^n$. Consider the problem

$$\begin{aligned} & \text{minimize} \quad \|\mathbf{x} - \mathbf{x}_0\| \\ & \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}. \end{aligned}$$

Show that this problem has a unique solution given by

$$\mathbf{x}^* = \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{b} + (\mathbf{I}_n - \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{A}) \mathbf{x}_0.$$

12.21 Given $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \geq n$, $\text{rank } \mathbf{A} = n$, and $\mathbf{b}_1, \dots, \mathbf{b}_p \in \mathbb{R}^m$, consider the problem

$$\text{minimize} \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}_1\|^2 + \|\mathbf{A}\mathbf{x} - \mathbf{b}_2\|^2 + \cdots + \|\mathbf{A}\mathbf{x} - \mathbf{b}_p\|^2.$$

Suppose that \mathbf{x}_i^* is a solution to the problem

$$\text{minimize} \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}_i\|^2,$$

where $i = 1, \dots, p$. Write the solution to the problem in terms of $\mathbf{x}_1^*, \dots, \mathbf{x}_p^*$.

12.22 Given $\mathbf{A} \in \mathbb{R}^{m \times n}$, $m \geq n$, $\text{rank } \mathbf{A} = n$, $\mathbf{b}_1, \dots, \mathbf{b}_p \in \mathbb{R}^m$, and $\alpha_1, \dots, \alpha_p \in \mathbb{R}$, consider the problem

$$\text{minimize } \alpha_1 \|\mathbf{Ax} - \mathbf{b}_1\|^2 + \alpha_2 \|\mathbf{Ax} - \mathbf{b}_2\|^2 + \cdots + \alpha_p \|\mathbf{Ax} - \mathbf{b}_p\|^2.$$

Suppose that \mathbf{x}_i^* is the solution to the problem

$$\text{minimize } \|\mathbf{Ax} - \mathbf{b}_i\|^2,$$

where $i = 1, \dots, p$. Assuming that $\alpha_1 + \cdots + \alpha_p > 0$, derive a simple expression for the solution to this problem in terms of $\mathbf{x}_1^*, \dots, \mathbf{x}_p^*$ and $\alpha_1, \dots, \alpha_p$.

12.23 Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $m \leq n$, and $\text{rank } \mathbf{A} = m$. Show that $\mathbf{x}^* = \mathbf{A}^\top (\mathbf{A}\mathbf{A}^\top)^{-1} \mathbf{b}$ is the only vector in $\mathcal{R}(\mathbf{A}^\top)$ satisfying $\mathbf{Ax}^* = \mathbf{b}$.

12.24 The purpose of this question is to derive a recursive least-squares algorithm where we *remove* (instead of add) a data point. To formulate the algorithm, suppose that we are given matrices \mathbf{A}_0 and \mathbf{A}_1 such that

$$\mathbf{A}_0 = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{a}_1^\top \end{bmatrix},$$

where $\mathbf{a}_1 \in \mathbb{R}^n$. Similarly, suppose that vectors $\mathbf{b}^{(0)}$ and $\mathbf{b}^{(1)}$ satisfy

$$\mathbf{b}^{(0)} = \begin{bmatrix} \mathbf{b}^{(1)} \\ b_1 \end{bmatrix},$$

where $b_1 \in \mathbb{R}$. Let $\mathbf{x}^{(0)}$ be the least-squares solution associated with $(\mathbf{A}_0, \mathbf{b}^{(0)})$ and $\mathbf{x}^{(1)}$ the least-squares solution associated with $(\mathbf{A}_1, \mathbf{b}^{(1)})$. Our goal is to write $\mathbf{x}^{(1)}$ in terms of $\mathbf{x}^{(0)}$ and the data point “removed,” (\mathbf{a}_1, b_1) . As usual, let \mathbf{G}_0 and \mathbf{G}_1 be the Grammians associated with $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$, respectively.

- a. Write down expressions for the least-squares solutions $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$ in terms of \mathbf{A}_0 , $\mathbf{b}^{(0)}$, \mathbf{A}_1 , and $\mathbf{b}^{(1)}$.
- b. Derive a formula for \mathbf{G}_1 in terms of \mathbf{G}_0 and \mathbf{a}_1 .
- c. Let $\mathbf{P}_0 = \mathbf{G}_0^{-1}$ and $\mathbf{P}_1 = \mathbf{G}_1^{-1}$. Derive a formula for \mathbf{P}_1 in terms of \mathbf{P}_0 and \mathbf{a}_1 . (The formula must not contain any matrix inversions.)
- d. Derive a formula for $\mathbf{A}_0^\top \mathbf{b}^{(0)}$ in terms of \mathbf{G}_1 , $\mathbf{x}^{(0)}$, and \mathbf{a}_1 .
- e. Finally, derive a formula for $\mathbf{x}^{(1)}$ in terms of $\mathbf{x}^{(0)}$, \mathbf{P}_1 , \mathbf{a}_1 , and b_1 . Use this and part c to write a recursive algorithm associated with successive removals of rows from $(\mathbf{A}_k, \mathbf{b}^{(k)})$.

12.25 Show that in Kaczmarz's algorithm, if $\mathbf{x}^{(0)} = \mathbf{0}$, then $\mathbf{x}^{(k)} \in \mathcal{R}(\mathbf{A}^\top)$ for all k .

12.26 Consider Kaczmarz's algorithm with $\mathbf{x}^{(0)} \neq \mathbf{0}$.

- a. Show that there exists a unique point minimizing $\|\mathbf{x} - \mathbf{x}^{(0)}\|$ subject to $\{\mathbf{x} : \mathbf{Ax} = \mathbf{b}\}$.
- b. Show that Kaczmarz's algorithm converges to the point in part a.

12.27 Consider Kaczmarz's algorithm with $\mathbf{x}^{(0)} = \mathbf{0}$, where $m = 1$; that is, $\mathbf{A} = [\mathbf{a}^\top] \in \mathbb{R}^{1 \times n}$ and $\mathbf{a} \neq \mathbf{0}$, and $0 < \mu < 2$. Show that there exists $0 \leq \gamma < 1$ such that $\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq \gamma \|\mathbf{x}^{(k)} - \mathbf{x}^*\|$ for all $k \geq 0$.

12.28 Show that in Kaczmarz's algorithm, if $\mu = 1$, then $b_{R(k)+1} - \mathbf{a}_{R(k)+1}^\top \mathbf{x}^{(k+1)} = 0$ for each k .

12.29 Consider the problem of minimizing $\|\mathbf{Ax} - \mathbf{b}\|^2$ over \mathbb{R}^n , where $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$. Let \mathbf{x}^* be a solution. Suppose that $\mathbf{A} = \mathbf{BC}$ is a full-rank factorization of \mathbf{A} ; that is, $\text{rank } \mathbf{A} = \text{rank } \mathbf{B} = \text{rank } \mathbf{C} = r$, and $\mathbf{B} \in \mathbb{R}^{m \times r}$, $\mathbf{C} \in \mathbb{R}^{r \times n}$. Show that the minimizer of $\|\mathbf{By} - \mathbf{b}\|$ over \mathbb{R}^r is \mathbf{Cx}^* .

12.30 Prove the following properties of generalized inverses:

- a. $(\mathbf{A}^\top)^\dagger = (\mathbf{A}^\dagger)^\top$.
- b. $(\mathbf{A}^\dagger)^\dagger = \mathbf{A}$.

12.31 Show that the Penrose definition of the generalized inverse is equivalent to Definition 12.1.

12.32 Construct matrices \mathbf{A}_1 and \mathbf{A}_2 such that $(\mathbf{A}_1 \mathbf{A}_2)^\dagger \neq \mathbf{A}_2^\dagger \mathbf{A}_1^\dagger$.

CHAPTER 13

UNCONSTRAINED OPTIMIZATION AND NEURAL NETWORKS

13.1 Introduction

In this chapter we apply the techniques of previous chapters to the training of feedforward neural networks. Neural networks have found numerous practical applications, ranging from telephone echo cancellation to aiding in the interpretation of EEG data (see, e.g., [108] and [72]). The essence of neural networks lies in the connection weights between neurons. The selection of these weights is referred to as *training* or *learning*. For this reason, we often refer to the weights as the *learning parameters*. A popular method for training a neural network is the *backpropagation algorithm*, based on an unconstrained optimization problem and an associated gradient algorithm applied to the problem. This chapter is devoted to a description of neural networks and the use of techniques developed in preceding chapters for the training of neural networks.

An *artificial neural network* is a circuit composed of interconnected simple circuit elements called *neurons*. Each neuron represents a map, typically with multiple inputs and a single output. Specifically, the output of the neuron is a function of the sum of the inputs, as illustrated in Figure 13.1. The

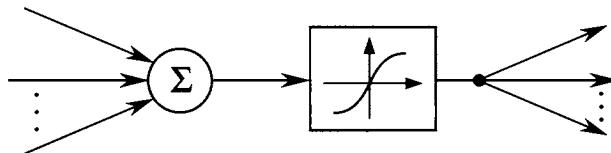


Figure 13.1 Single neuron.

function at the output of the neuron is called the *activation function*. We use the symbol shown in Figure 13.2 to represent a single neuron. Note that the single output of the neuron may be used as an input to several other neurons, and therefore the symbol for a single neuron has multiple arrows emanating from it. A neural network may be implemented using an analog circuit. In this case inputs and outputs may be represented by currents and voltages.

A neural network consists of interconnected neurons, with the inputs to each neuron consisting of weighted outputs of other neurons. The interconnections allow exchange of data or information between neurons. In a *feed-forward neural network*, the neurons are interconnected in layers, so that the data flow in only one direction. Thus, each neuron receives information only from neurons in the preceding layer: The inputs to each neuron are weighted outputs of neurons in the preceding layer. Figure 13.3 illustrates the structure of feedforward neural networks. The first layer in the network is called the *input layer*, and the last layer is called the *output layer*. The layers in between the input and output layers are called *hidden layers*.

We can view a neural network as simply a particular implementation of a map from \mathbb{R}^n to \mathbb{R}^m , where n is the number of inputs x_1, \dots, x_n and m is the number of outputs y_1, \dots, y_m . The map that is implemented by a neural network depends on the weights of the interconnections in the network. Therefore, we can change the mapping that is implemented by the network by adjusting the values of the weights in the network. The information about the mapping is “stored” in the weights over all the neurons, and thus the neural network is a *distributed* representation of the mapping. Moreover, for any given input, computation of the corresponding output is achieved through the collective effect of individual input-output characteristics of each neuron; therefore, the neural network can be considered as a *parallel* computation device. We point out that the ability to implement or approximate a map encompasses many important practical applications. For example, pattern

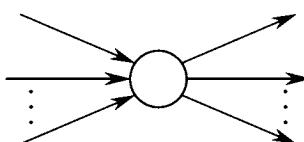


Figure 13.2 Symbol for a single neuron.

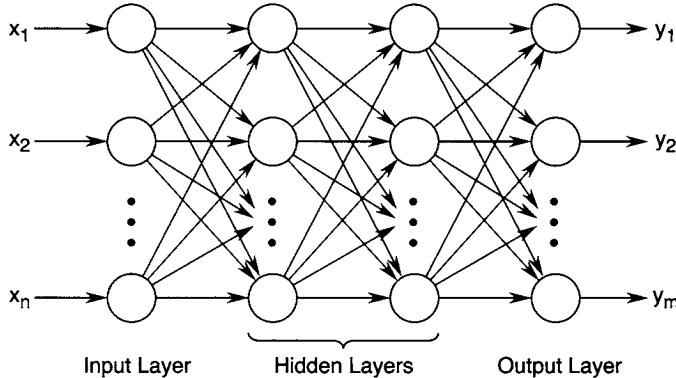


Figure 13.3 Structure of a feedforward neural network.

recognition and classification problems can be viewed as function implementation or approximation problems.

Suppose that we are given a map $\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ that we wish to implement using a given neural network. Our task boils down to selecting the interconnection weights in the network appropriately. As mentioned earlier, we refer to this task as *training* of the neural network or *learning* by the neural network. We use input-output examples of the given map to train the neural network. Specifically, let $(\mathbf{x}_{d,1}, \mathbf{y}_{d,1}), \dots, (\mathbf{x}_{d,p}, \mathbf{y}_{d,p}) \in \mathbb{R}^n \times \mathbb{R}^m$, where each $\mathbf{y}_{d,i}$ is the output of the map \mathbf{F} corresponding to the input $\mathbf{x}_{d,i}$; that is, $\mathbf{y}_{d,i} = \mathbf{F}(\mathbf{x}_{d,i})$. We refer to the set $\{(\mathbf{x}_{d,1}, \mathbf{y}_{d,1}), \dots, (\mathbf{x}_{d,p}, \mathbf{y}_{d,p})\}$ as the *training set*. We train the neural network by adjusting the weights such that the map that is implemented by the network is close to the desired map \mathbf{F} . For this reason, we can think of neural networks as function approximators.

The form of learning described above can be thought of as learning with a teacher. The teacher supplies questions to the network in the form of $\mathbf{x}_{d,1}, \dots, \mathbf{x}_{d,p}$ and tells the network the correct answers $\mathbf{y}_{d,1}, \dots, \mathbf{y}_{d,p}$. Training of the network then comprises applying a training algorithm that adjusts weights based on the error between the computed and desired outputs; that is, the difference between $\mathbf{y}_{d,i} = \mathbf{F}(\mathbf{x}_{d,i})$ and the output of the neural network corresponding to $\mathbf{x}_{d,i}$. Having trained the neural network, our hope is that the network correctly generalizes the examples used in the training set. By this we mean that the network should correctly implement the mapping \mathbf{F} and produce the correct output corresponding to any input, including those that were not a part of the training set.

As we shall see in the remainder of this chapter, the training problem can be formulated as an optimization problem. We can then use optimization techniques and search methods (e.g., steepest descent, conjugate gradients [69], and quasi-Newton) for selection of the weights. The training algorithms are based on such optimization algorithms.

In the literature, for obvious reasons, the form of learning described above is referred to as *supervised learning*, a term which suggests that there is also a form of learning called *unsupervised learning*. Indeed, this is the case. However, unsupervised learning does not fit into the framework described above. Therefore, we do not discuss the idea of unsupervised learning any further. We refer the interested reader to [60].

13.2 Single-Neuron Training

Consider a single neuron, as shown in Figure 13.4. For this particular neuron, the activation function is simply the identity (linear function with unit slope). The neuron implements the following (linear) map from \mathbb{R}^n to \mathbb{R} :

$$y = \sum_{i=1}^n w_i x_i = \mathbf{x}^\top \mathbf{w},$$

where $\mathbf{x} = [x_1, \dots, x_n]^\top \in \mathbb{R}^n$ is the vector of inputs, $y \in \mathbb{R}$ is the output, and $\mathbf{w} = [w_1, \dots, w_n]^\top \in \mathbb{R}^n$ is the vector of weights. Suppose that we are given a map $F : \mathbb{R}^n \rightarrow \mathbb{R}$. We wish to find the value of the weights w_1, \dots, w_n such that the neuron approximates the map F as closely as possible. To do this, we use a training set consisting of p pairs $\{(\mathbf{x}_{d,1}, y_{d,1}), \dots, (\mathbf{x}_{d,p}, y_{d,p})\}$, where $\mathbf{x}_{d,i} \in \mathbb{R}^n$ and $y_{d,i} \in \mathbb{R}$, $i = 1, \dots, p$. For each i , $y_{d,i} = F(\mathbf{x}_{d,i})$ is the “desired” output corresponding to the given input $\mathbf{x}_{d,i}$. The training problem can then be formulated as the following optimization problem:

$$\text{minimize } \frac{1}{2} \sum_{i=1}^p (y_{d,i} - \mathbf{x}_{d,i}^\top \mathbf{w})^2,$$

where the minimization is taken over all $\mathbf{w} = [w_1, \dots, w_n]^\top \in \mathbb{R}^n$. Note that the objective function represents the sum of the squared errors between the desired outputs $y_{d,i}$ and the corresponding outputs of the neuron $\mathbf{x}_{d,i}^\top \mathbf{w}$. The factor of $1/2$ is added for notational convenience and does not change the minimizer.

The objective function above can be written in matrix form as follows. First define the matrix $\mathbf{X}_d \in \mathbb{R}^{n \times p}$ and vector $\mathbf{y}_d \in \mathbb{R}^p$ by

$$\mathbf{X}_d = [\mathbf{x}_{d,1} \cdots \mathbf{x}_{d,p}],$$

$$\mathbf{y}_d = \begin{bmatrix} y_{d,1} \\ \vdots \\ y_{d,p} \end{bmatrix}.$$

Then, the optimization problem becomes

$$\text{minimize } \frac{1}{2} \|\mathbf{y}_d - \mathbf{X}_d^\top \mathbf{w}\|^2.$$

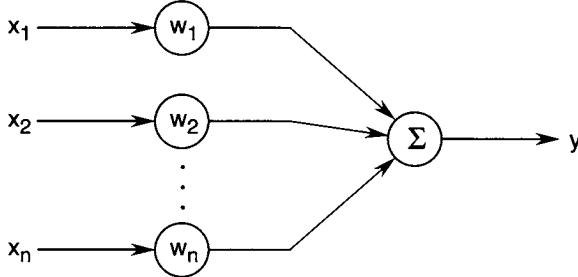


Figure 13.4 Single linear neuron.

There are two cases to consider in this optimization problem: $p \leq n$ and $p > n$. We first consider the case where $p \leq n$, that is, where we have at most as many training pairs as the number of weights. For convenience, we assume that $\text{rank } \mathbf{X}_d^\top = p$. In this case there are an infinitely many points satisfying $\mathbf{y}_d = \mathbf{X}_d^\top \mathbf{w}$. Hence, there are infinitely many solutions to the optimization problem above, with the optimal objective function value of 0. Therefore, we have a choice of which optimal solution to select. A possible criterion for this selection is that of minimizing the solution norm. This is exactly the problem considered in Section 12.3. Recall that the minimum-norm solution is $\mathbf{w}^* = \mathbf{X}_d(\mathbf{X}_d^\top \mathbf{X}_d)^{-1} \mathbf{y}_d$. An efficient iterative algorithm for finding this solution is Kaczmarz's algorithm (discussed in Section 12.4). Kaczmarz's algorithm in this setting takes the form

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mu \frac{e_k \mathbf{x}_{d,R(k)+1}}{\|\mathbf{x}_{d,R(k)+1}\|^2},$$

where $\mathbf{w}^{(0)} = \mathbf{0}$ and

$$e_k = y_{d,R(k)+1} - \mathbf{x}_{d,R(k)+1}^\top \mathbf{w}^{(k)}.$$

Recall that $R(k)$ is the unique integer in $\{0, \dots, p-1\}$ satisfying $k = lp + R(k)$ for some integer l ; that is, $R(k)$ is the remainder that results if we divide k by p (see Section 12.4 for further details on the algorithm).

The algorithm above was applied to the training of linear neurons by Widrow and Hoff (see [132] for some historical remarks). The single neuron together with the training algorithm above is illustrated in Figure 13.5 and is often called *Adaline*, an acronym for *adaptive linear element*.

We now consider the case where $p > n$. Here, we have more training points than the number of weights. We assume that $\text{rank } \mathbf{X}_d^\top = n$. In this case the objective function $\frac{1}{2} \|\mathbf{y}_d - \mathbf{X}_d^\top \mathbf{w}\|^2$ is simply a strictly convex quadratic function of \mathbf{w} , because the matrix $\mathbf{X}_d \mathbf{X}_d^\top$ is a positive definite matrix. To solve this optimization problem, we have at our disposal the whole slew of unconstrained optimization algorithms considered in earlier chapters. For example, we can use a gradient algorithm, which in this case takes the form

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \alpha_k \mathbf{X}_d e^{(k)},$$

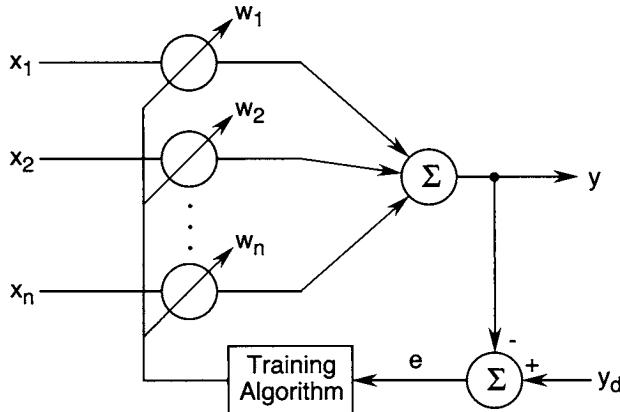


Figure 13.5 Adaline.

where $e^{(k)} = y_d - \mathbf{X}_d^\top \mathbf{w}^{(k)}$.

The discussion above assumed that the activation function for the neuron is the identity map. The derivation and analysis of the algorithms can be extended to the case of a general differentiable activation function f_a . Specifically, the output of the neuron in this case is given by

$$y = f_a \left(\sum_{i=1}^n w_i x_i \right) = f_a (\mathbf{x}^\top \mathbf{w}).$$

The algorithm for the case of a single training pair (\mathbf{x}_d, y_d) has the form

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mu \frac{e_k \mathbf{x}_d}{\|\mathbf{x}_d\|^2},$$

where the error is given by

$$e_k = y_d - f_a (\mathbf{x}_d^\top \mathbf{w}^{(k)}).$$

For a convergence analysis of the algorithm above, see [64].

13.3 The Backpropagation Algorithm

In Section 13.2 we considered the problem of training a single neuron. In this section we consider a neural network consisting of many layers. For simplicity of presentation, we restrict our attention to networks with three layers, as depicted in Figure 13.6. The three layers are referred to as the input, hidden, and output layers. There are n inputs x_i , where $i = 1, \dots, n$. We have m outputs y_s , $s = 1, \dots, m$. There are l neurons in the hidden layer. The outputs

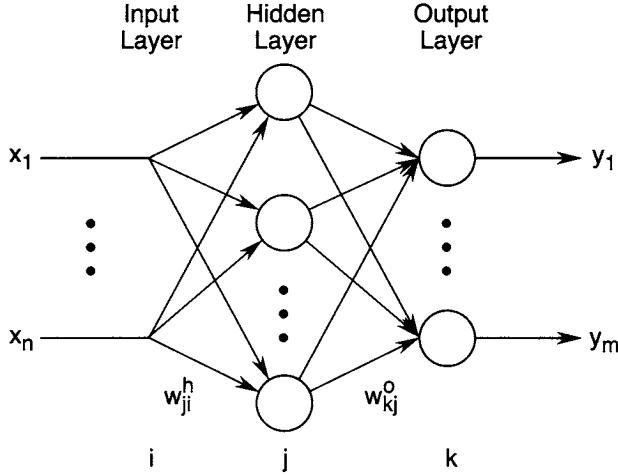


Figure 13.6 Three-layered neural network.

of the neurons in the hidden layer are z_j , where $j = 1, \dots, l$. The inputs x_1, \dots, x_n are distributed to the neurons in the hidden layer. We may think of the neurons in the input layer as single-input-single-output linear elements, with each activation function being the identity map. In Figure 13.6 we do not explicitly depict the neurons in the input layer; instead, we illustrate the neurons as signal splitters. We denote the activation functions of the neurons in the hidden layer by f_j^h , where $j = 1, \dots, l$, and the activation functions of the neurons in the output layer by f_s^o , where $s = 1, \dots, m$. Note that each activation function is a function from \mathbb{R} to \mathbb{R} .

We denote the weights for inputs into the hidden layer by w_{ji}^h , $i = 1, \dots, n$, $j = 1, \dots, l$. We denote the weights for inputs from the hidden layer into the output layer by w_{sj}^o , $j = 1, \dots, l$, $s = 1, \dots, m$. Given the weights w_{ji}^h and w_{sj}^o , the neural network implements a map from \mathbb{R}^n to \mathbb{R}^m . To find an explicit formula for this map, let us denote the input to the j th neuron in the hidden layer by v_j and the output of the j th neuron in the hidden layer by z_j . Then, we have

$$v_j = \sum_{i=1}^n w_{ji}^h x_i,$$

$$z_j = f_j^h \left(\sum_{i=1}^n w_{ji}^h x_i \right).$$

The output from the s th neuron of the output layer is

$$y_s = f_s^o \left(\sum_{j=1}^l w_{sj}^o z_j \right).$$

Therefore, the relationship between the inputs x_i , $i = 1, \dots, n$, and the s th output y_s is given by

$$\begin{aligned} y_s &= f_s^o \left(\sum_{j=1}^l w_{sj}^o f_j^h(v_j) \right) \\ &= f_s^o \left(\sum_{j=1}^l w_{sj}^o f_j^h \left(\sum_{i=1}^n w_{ji}^h x_i \right) \right) \\ &= F_s(x_1, \dots, x_n). \end{aligned}$$

The overall mapping that the neural network implements is therefore given by

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} F_1(x_1, \dots, x_n) \\ \vdots \\ F_m(x_1, \dots, x_n) \end{bmatrix}.$$

We now consider the problem of training the neural network. As for the single neuron considered in Section 13.2, we analyze the case where the training set consists of a single pair $(\mathbf{x}_d, \mathbf{y}_d)$, where $\mathbf{x}_d \in \mathbb{R}^n$ and $\mathbf{y}_d \in \mathbb{R}^m$. In practice, the training set consists of many such pairs, and training is typically performed with each pair at a time (see, e.g., [65] or [113]). Our analysis is therefore also relevant to the general training problem with multiple training pairs.

The training of the neural network involves adjusting the weights of the network such that the output generated by the network for the given input $\mathbf{x}_d = [x_{d1}, \dots, x_{dn}]^\top$ is as close to \mathbf{y}_d as possible. Formally, this can be formulated as the following optimization problem:

$$\text{minimize } \frac{1}{2} \sum_{s=1}^m (y_{ds} - y_s)^2,$$

where y_s , $s = 1, \dots, m$, are the actual outputs of the neural network in response to the inputs x_{d1}, \dots, x_{dn} , as given by

$$y_s = f_s^o \left(\sum_{j=1}^l w_{sj}^o f_j^h \left(\sum_{i=1}^n w_{ji}^h x_i \right) \right).$$

This minimization is taken over all w_{ji}^h , w_{sj}^o , $i = 1, \dots, n$, $j = 1, \dots, l$, $s = 1, \dots, m$. For simplicity of notation, we use the symbol \mathbf{w} for the vector

$$\mathbf{w} = \{w_{ji}^h, w_{sj}^o : i = 1, \dots, n, j = 1, \dots, l, s = 1, \dots, m\}$$

and the symbol E for the objective function to be minimized; that is,

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2} \sum_{s=1}^m (y_{ds} - y_s)^2 \\ &= \frac{1}{2} \sum_{s=1}^m \left(y_{ds} - f_s^o \left(\sum_{j=1}^l w_{sj}^o f_j^h \left(\sum_{i=1}^n w_{ji}^h x_{di} \right) \right) \right)^2. \end{aligned}$$

To solve the optimization problem above, we use a gradient algorithm with fixed step size. To formulate the algorithm, we need to compute the partial derivatives of E with respect to each component of \mathbf{w} . For this, let us first fix the indices i , j , and s . We first compute the partial derivative of E with respect to w_{sj}^o . For this, we write

$$E(\mathbf{w}) = \frac{1}{2} \sum_{p=1}^m \left(y_{dp} - f_p^o \left(\sum_{q=1}^l w_{pq}^o z_q \right) \right)^2,$$

where for each $q = 1, \dots, l$,

$$z_q = f_q^h \left(\sum_{i=1}^n w_{qi}^h x_{di} \right).$$

Using the chain rule, we obtain

$$\frac{\partial E}{\partial w_{sj}^o}(\mathbf{w}) = -(y_{ds} - y_s) f_s^{o'} \left(\sum_{q=1}^l w_{sq}^o z_q \right) z_j,$$

where $f_s^{o'} : \mathbb{R} \rightarrow \mathbb{R}$ is the derivative of f_s^o . For simplicity of notation, we write

$$\delta_s = (y_{ds} - y_s) f_s^{o'} \left(\sum_{q=1}^l w_{sq}^o z_q \right).$$

We can think of each δ_s as a scaled output error, because it is the difference between the actual output y_s of the neural network and the desired output y_{ds} , scaled by $f_s^{o'} \left(\sum_{q=1}^l w_{sq}^o z_q \right)$. Using the δ_s notation, we have

$$\frac{\partial E}{\partial w_{sj}^o}(\mathbf{w}) = -\delta_s z_j.$$

We next compute the partial derivative of E with respect to w_{ji}^h . We start with the equation

$$E(\mathbf{w}) = \frac{1}{2} \sum_{p=1}^m \left(y_{dp} - f_p^o \left(\sum_{q=1}^l w_{pq}^o f_q^h \left(\sum_{r=1}^n w_{qr}^h x_{dr} \right) \right) \right)^2.$$

Using the chain rule once again, we get

$$\frac{\partial E}{\partial w_{ji}^h}(\mathbf{w}) = - \sum_{p=1}^m (y_{dp} - y_p) f_p^{o'} \left(\sum_{q=1}^l w_{pq}^o z_q \right) w_{pj}^o f_j^{h'} \left(\sum_{r=1}^n w_{jr}^h x_{dr} \right) x_{di},$$

where $f_j^{h'} : \mathbb{R} \rightarrow \mathbb{R}$ is the derivative of f_j^h . Simplifying the above yields

$$\frac{\partial E}{\partial w_{ji}^h}(\mathbf{w}) = - \left(\sum_{p=1}^m \delta_p w_{pj}^o \right) f_j^{h'}(v_j) x_{di}.$$

We are now ready to formulate the gradient algorithm for updating the weights of the neural network. We write the update equations for the two sets of weights w_{sj}^o and w_{ji}^h separately. We have

$$\begin{aligned} w_{sj}^{o(k+1)} &= w_{sj}^{o(k)} + \eta \delta_s^{(k)} z_j^{(k)}, \\ w_{ji}^{h(k+1)} &= w_{ji}^{h(k)} + \eta \left(\sum_{p=1}^m \delta_p^{(k)} w_{pj}^{o(k)} \right) f_j^{h'}(v_j^{(k)}) x_{di}, \end{aligned}$$

where η is the (fixed) step size and

$$\begin{aligned} v_j^{(k)} &= \sum_{i=1}^n w_{ji}^{h(k)} x_{di}, \\ z_j^{(k)} &= f_j^h(v_j^{(k)}), \\ y_s^{(k)} &= f_s^o \left(\sum_{q=1}^l w_{sq}^{o(k)} z_q^{(k)} \right), \\ \delta_s^{(k)} &= (y_{ds} - y_s^{(k)}) f_s^{o'} \left(\sum_{q=1}^l w_{sq}^{o(k)} z_q^{(k)} \right). \end{aligned}$$

The update equation for the weights w_{sj}^o of the output layer neurons is illustrated in Figure 13.7, whereas the update equation for the weights w_{ji}^h of the hidden layer neurons is illustrated in Figure 13.8.

The update equations above are referred to in the literature as the *backpropagation algorithm*. The reason for the name *backpropagation* is that the output errors $\delta_1^{(k)}, \dots, \delta_m^{(k)}$ are propagated back from the output layer to the hidden layer and are used in the update equation for the hidden layer weights, as illustrated in Figure 13.8. In the discussion above we assumed only a single hidden layer. In general, we may have multiple hidden layers—in this case the update equations for the weights will resemble the equations derived above. In the general case the output errors are propagated backward from layer to layer and are used to update the weights at each layer.

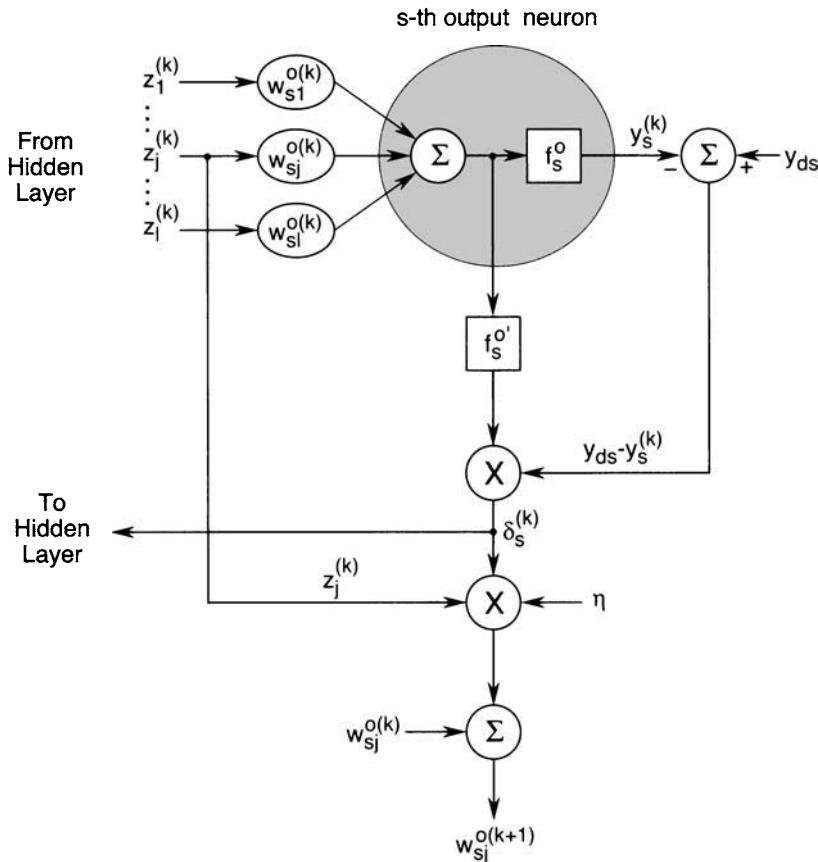


Figure 13.7 Illustration of the update equation for the output layer weights.

We summarize the backpropagation algorithm qualitatively as follows. Using the inputs x_{di} and the current set of weights, we first compute the quantities $v_j^{(k)}$, $z_j^{(k)}$, $y_s^{(k)}$, and $\delta_s^{(k)}$, in turn. This is called the *forward pass* of the algorithm, because it involves propagating the input forward from the input layer to the output layer. Next, we compute the updated weights using the quantities computed in the forward pass. This is called the *reverse pass* of the algorithm, because it involves propagating the computed output errors $\delta_s^{(k)}$ backward through the network. We illustrate the backpropagation procedure numerically in the following example.

Example 13.1 Consider the simple feedforward neural network shown in Figure 13.9. The activation functions for all the neurons are given by $f(v) = 1/(1 + e^{-v})$. This particular activation function has the convenient property

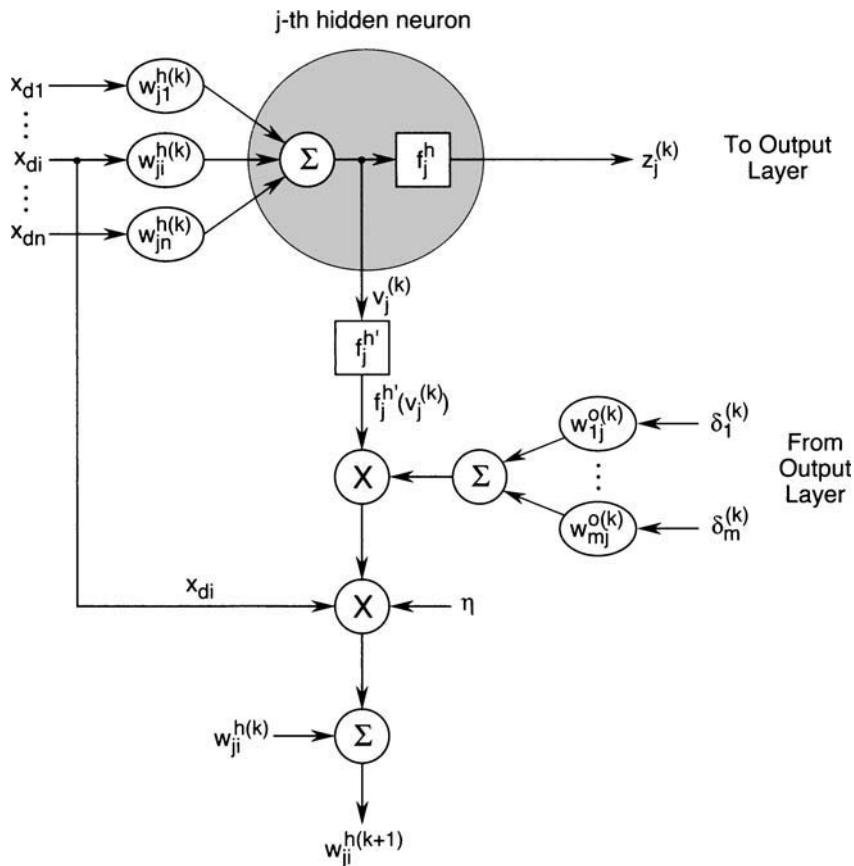


Figure 13.8 Illustration of the update equation for the hidden layer weights.

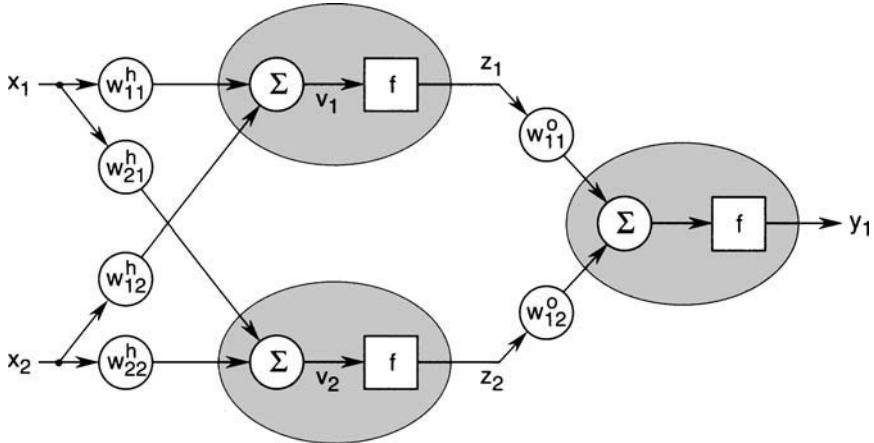


Figure 13.9 Neural network for Example 13.1.

that $f'(v) = f(v)(1 - f(v))$. Therefore, using this property, we can write

$$\begin{aligned}\delta_1 &= (y_d - y_1)f' \left(\sum_{q=1}^2 w_{1q}^o z_q \right) \\ &= (y_d - y_1)f \left(\sum_{q=1}^2 w_{1q}^o z_q \right) \left(1 - f \left(\sum_{q=1}^2 w_{1q}^o z_q \right) \right) \\ &= (y_d - y_1)y_1(1 - y_1).\end{aligned}$$

Suppose that the initial weights are $w_{11}^{h(0)} = 0.1$, $w_{12}^{h(0)} = 0.3$, $w_{21}^{h(0)} = 0.3$, $w_{22}^{h(0)} = 0.4$, $w_{11}^{o(0)} = 0.4$, and $w_{12}^{o(0)} = 0.6$. Let $\mathbf{x}_d = [0.2, 0.6]^\top$ and $y_d = 0.7$. Perform one iteration of the backpropagation algorithm to update the weights of the network. Use a step size of $\eta = 10$.

To proceed, we first compute

$$\begin{aligned}v_1^{(0)} &= w_{11}^{h(0)}x_{d1} + w_{12}^{h(0)}x_{d2} = 0.2, \\ v_2^{(0)} &= w_{21}^{h(0)}x_{d1} + w_{22}^{h(0)}x_{d2} = 0.3.\end{aligned}$$

Next, we compute

$$\begin{aligned}z_1^{(0)} &= f(v_1^{(0)}) = \frac{1}{1 + e^{-0.2}} = 0.5498, \\ z_2^{(0)} &= f(v_2^{(0)}) = \frac{1}{1 + e^{-0.3}} = 0.5744.\end{aligned}$$

We then compute

$$y_1^{(0)} = f \left(w_{11}^{o(0)}z_1^{(0)} + w_{12}^{o(0)}z_2^{(0)} \right) = f(0.5646) = 0.6375,$$

which gives an output error of

$$\delta_1^{(0)} = (y_d - y_1^{(0)})y_1^{(0)}(1 - y_1^{(0)}) = 0.01444.$$

This completes the forward pass.

To update the weights, we use

$$\begin{aligned} w_{11}^{o(1)} &= w_{11}^{o(0)} + \eta\delta_1^{(0)}z_1^{(0)} = 0.4794, \\ w_{12}^{o(1)} &= w_{12}^{o(0)} + \eta\delta_1^{(0)}z_2^{(0)} = 0.6830, \end{aligned}$$

and, using the fact that $f'(v_j^{(0)}) = f(v_j^{(0)})(1 - f(v_j^{(0)})) = z_j^{(0)}(1 - z_j^{(0)})$, we get

$$\begin{aligned} w_{11}^{h(1)} &= w_{11}^{h(0)} + \eta\delta_1^{(0)}w_{11}^{o(0)}z_1^{(0)}(1 - z_1^{(0)})x_{d1} = 0.1029, \\ w_{12}^{h(1)} &= w_{12}^{h(0)} + \eta\delta_1^{(0)}w_{11}^{o(0)}z_1^{(0)}(1 - z_1^{(0)})x_{d2} = 0.3086, \\ w_{21}^{h(1)} &= w_{21}^{h(0)} + \eta\delta_1^{(0)}w_{12}^{o(0)}z_2^{(0)}(1 - z_2^{(0)})x_{d1} = 0.3042, \\ w_{22}^{h(1)} &= w_{22}^{h(0)} + \eta\delta_1^{(0)}w_{12}^{o(0)}z_2^{(0)}(1 - z_2^{(0)})x_{d2} = 0.4127. \end{aligned}$$

Thus, we have completed one iteration of the backpropagation algorithm. We can easily check that $y_1^{(1)} = 0.6588$, and hence $|y_d - y_1^{(1)}| < |y_d - y_1^{(0)}|$; that is, the actual output of the neural network has become closer to the desired output as a result of updating the weights.

After 15 iterations of the backpropagation algorithm, we get

$$\begin{aligned} w_{11}^{o(15)} &= 0.6365, \\ w_{12}^{o(15)} &= 0.8474, \\ w_{11}^{h(15)} &= 0.1105, \\ w_{12}^{h(15)} &= 0.3315, \\ w_{21}^{h(15)} &= 0.3146, \\ w_{22}^{h(15)} &= 0.4439. \end{aligned}$$

The resulting value of the output corresponding to the input $\mathbf{x}_d = [0.2, 0.6]^\top$ is $y_1^{(15)} = 0.6997$. ■

In the example above, we considered an activation function of the form

$$f(v) = \frac{1}{1 + e^{-v}}.$$

This function is called a *sigmoid* and is a popular activation function used in practice. The sigmoid function is illustrated in Figure 13.10. It is possible to use a more general version of the sigmoid function, of the form

$$g(v) = \frac{\beta}{1 + e^{-(v-\theta)}}.$$

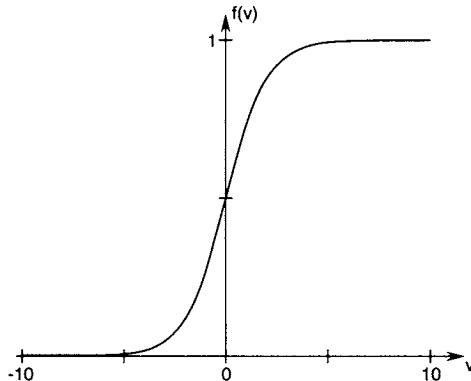


Figure 13.10 Sigmoid function.

The parameters β and θ represent *scale* and *shift* (or *location*) *parameters* respectively. The parameter θ is often interpreted as a threshold. If such an activation function is used in a neural network, we would also want to adjust the values of the parameters β and θ , which also affect the value of the objective function to be minimized. However, it turns out that these parameters can be incorporated into the backpropagation algorithm simply by treating them as additional weights in the network. Specifically, we can represent a neuron with activation function g as one with activation function f with the addition of two extra weights, as shown in Figure 13.11.

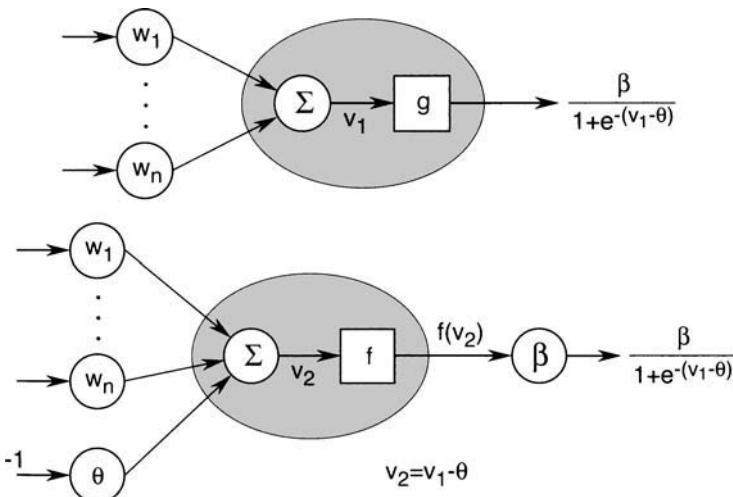


Figure 13.11 Two configurations that are equivalent.

Example 13.2 Consider the same neural network as in Example 13.1. We introduce shift parameters θ_1 , θ_2 , and θ_3 to the activation functions in the neurons. Using the configuration illustrated in Figure 13.11, we can incorporate the shift parameters into the backpropagation algorithm. We have

$$\begin{aligned} v_1 &= w_{11}^h x_{d1} + w_{12}^h x_{d2} - \theta_1, \\ v_2 &= w_{21}^h x_{d1} + w_{22}^h x_{d2} - \theta_2, \\ z_1 &= f(v_1), \\ z_2 &= f(v_2), \\ y_1 &= f(w_{11}^o z_1 + w_{12}^o z_2 - \theta_3), \\ \delta_1 &= (y_d - y_1)y_1(1 - y_1), \end{aligned}$$

where f is the sigmoid function:

$$f(v) = \frac{1}{1 + e^{-v}}.$$

The components of the gradient of the objective function E with respect to the shift parameters are

$$\begin{aligned} \frac{\partial E}{\partial \theta_1}(\mathbf{w}) &= \delta_1 w_{11}^o z_1 (1 - z_1), \\ \frac{\partial E}{\partial \theta_2}(\mathbf{w}) &= \delta_1 w_{12}^o z_2 (1 - z_2), \\ \frac{\partial E}{\partial \theta_3}(\mathbf{w}) &= \delta_1. \end{aligned}$$

■

In the next example, we apply the network discussed in Example 13.2 to solve the celebrated *exclusive OR (XOR) problem* (see [113]).

Example 13.3 Consider the neural network of Example 13.2. We wish to train the neural network to approximate the *exclusive OR (XOR)* function, defined in Table 13.1. Note that the XOR function has two inputs and one output.

To train the neural network, we use the following training pairs:

$$\begin{aligned} \mathbf{x}_{d,1} &= [0, 0]^\top, & y_{d,1} &= 0, \\ \mathbf{x}_{d,2} &= [0, 1]^\top, & y_{d,2} &= 1, \\ \mathbf{x}_{d,3} &= [1, 0]^\top, & y_{d,3} &= 1, \\ \mathbf{x}_{d,4} &= [1, 1]^\top, & y_{d,4} &= 0. \end{aligned}$$

We now apply the backpropagation algorithm to train the network using the training pairs above. To do this, we apply one pair per iteration in a cyclic

Table 13.1 Truth Table for XOR Function

x_1	x_2	$F(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

Table 13.2 Response of the Trained Network of Example 13.3

x_1	x_2	y_1
0	0	0.007
0	1	0.99
1	0	0.99
1	1	0.009

fashion. In other words, in the k th iteration of the algorithm, we apply the pair $(\mathbf{x}_{d,R(k)+1}, y_{d,R(k)+1})$, where, as in Kaczmarz's algorithm, $R(k)$ is the unique integer in $\{0, 1, 2, 3\}$ satisfying $k = 4l + R(k)$ for some integer l ; that is, $R(k)$ is the remainder that results if we divide k by 4 (see Section 12.4).

The experiment yields the following weights (see Exercise 13.5):

$$\begin{aligned} w_{11}^o &= -11.01, \\ w_{12}^o &= 10.92, \\ w_{11}^h &= -7.777, \\ w_{12}^h &= -8.403, \\ w_{21}^h &= -5.593, \\ w_{22}^h &= -5.638, \\ \theta_1 &= -3.277, \\ \theta_2 &= -8.357, \\ \theta_3 &= 5.261. \end{aligned}$$

Table 13.2 shows the output of the neural network with the weights above corresponding to the training input data. Figure 13.12 shows a plot of the function that is implemented by this neural network. ■

For a more comprehensive treatment of neural networks, see [58], [59], or [137]. For applications of neural networks to optimization, signal processing, and control problems, see [28] and [67].

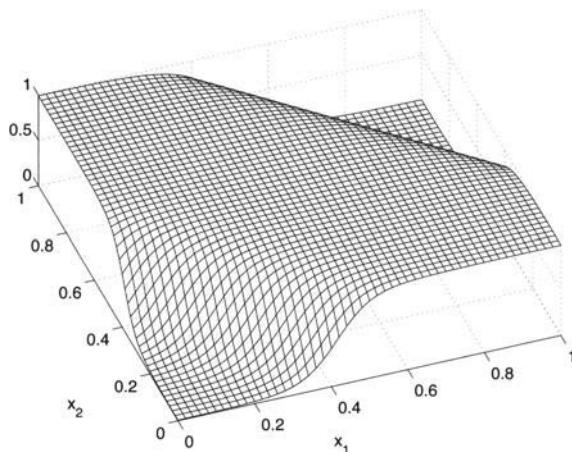


Figure 13.12 Plot of the function implemented by the trained network of Example 13.3.

EXERCISES

13.1 Consider a single linear neuron, with n inputs (see Figure 13.4). Suppose that we are given $\mathbf{X}_d \in \mathbb{R}^{n \times p}$ and $\mathbf{y}_d \in \mathbb{R}^p$ representing p training pairs, where $p > n$. The objective function to be minimized in the training of the neuron is

$$f(\mathbf{w}) = \frac{1}{2} \|\mathbf{y}_d - \mathbf{X}_d^\top \mathbf{w}\|^2.$$

- a. Find the gradient of the objective function.
- b. Write the conjugate gradient algorithm for training the neuron.
- c. Suppose that we wish to approximate the function $F : \mathbb{R}^2 \rightarrow \mathbb{R}$ given by

$$F(\mathbf{x}) = (\sin x_1)(\cos x_2).$$

Use the conjugate gradient algorithm from part b to train the linear neuron, using the following training points:

$$\{\mathbf{x} : x_1, x_2 = -0.5, 0, 0.5\}.$$

It may helpful to use the MATLAB program from Exercise 10.11.

- d. Plot the level sets of the objective function for the problem in part c, at levels 0.01, 0.1, 0.2, and 0.4. Check if the solution in part c agrees with the level sets.

- e. Plot the error function $e(\mathbf{x}) = F(\mathbf{x}) - \mathbf{w}^{*\top} \mathbf{x}$ versus x_1 and x_2 , where \mathbf{w}^* is the optimal weight vector obtained in part c.

13.2 Consider the Adaline, depicted in Figure 13.5. Assume that we have a single training pair (\mathbf{x}_d, y_d) , where $\mathbf{x}_d \neq \mathbf{0}$. Suppose that we use the Widrow-Hoff algorithm to adjust the weights:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mu \frac{e_k \mathbf{x}_d}{\mathbf{x}_d^\top \mathbf{x}_d},$$

where $e_k = y_d - \mathbf{x}_d^\top \mathbf{w}^{(k)}$.

- a. Write an expression for e_{k+1} as a function of e_k and μ .
- b. Find the largest range of values for μ for which $e_k \rightarrow 0$ (for any initial condition $\mathbf{w}^{(0)}$).

13.3 As in Exercise 13.2, consider the Adaline. Consider the case in which there are multiple pairs in the training set $\{(\mathbf{x}_{d,1}, y_{d,1}), \dots, (\mathbf{x}_{d,p}, y_{d,p})\}$, where $p \leq n$ and $\text{rank } \mathbf{X}_d = p$ (the matrix \mathbf{X}_d has $\mathbf{x}_{d,i}$ as its i th column). Suppose that we use the following training algorithm:

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mathbf{X}_d (\mathbf{X}_d^\top \mathbf{X}_d)^{-1} \boldsymbol{\mu} \mathbf{e}^{(k)},$$

where $\mathbf{e}^{(k)} = \mathbf{y}_d - \mathbf{X}_d^\top \mathbf{w}^{(k)}$ and $\boldsymbol{\mu}$ is a given constant $p \times p$ matrix.

- a. Find an expression for $\mathbf{e}^{(k+1)}$ as a function of $\mathbf{e}^{(k)}$ and $\boldsymbol{\mu}$.
- b. Find a necessary and sufficient condition on $\boldsymbol{\mu}$ for which $\mathbf{e}^{(k)} \rightarrow \mathbf{0}$ (for any initial condition $\mathbf{w}^{(0)}$).

13.4 Consider the three-layered neural network described in Example 13.1 (see Figure 13.9). Implement the backpropagation algorithm for this network in MATLAB. Test the algorithm for the training pair $\mathbf{x}_d = [0, 1]^\top$ and $y_d = 1$. Use a step size of $\eta = 50$ and initial weights as in the Example 13.1.

13.5 Consider the neural network of Example 13.3, with training pairs for the XOR problem. Use MATLAB to implement the training algorithm described in Example 13.3, with a step size of $\eta = 10.0$. Tabulate the outputs of the trained network corresponding to the training input data.

CHAPTER 14

GLOBAL SEARCH ALGORITHMS

14.1 Introduction

The iterative algorithms in previous chapters, in particular gradient methods, Newton's method, conjugate gradient methods, and quasi-Newton methods, start with an initial point and then generate a sequence of iterates. Typically, the best we can hope for is that the sequence converges to a local minimizer. For this reason, it is often desirable for the initial point to be close to a global minimizer. Moreover, these methods require first derivatives (and also second derivatives in the case of Newton's method).

In this chapter we discuss various search methods that are global in nature in the sense that they attempt to search throughout the entire feasible set. These methods use only objective function values and do not require derivatives. Consequently, they are applicable to a much wider class of optimization problems. In some cases, they can also be used to generate “good” initial (starting) points for the iterative methods discussed in earlier chapters. Some of the methods we discuss in this chapter (specifically, the randomized search methods) are also used in combinatorial optimization, where the feasible set is finite (discrete), but typically large.

14.2 The Nelder-Mead Simplex Algorithm

The method originally proposed by Spendley, Hext, and Hinsworth [122] in 1962 was improved by Nelder and Mead [97] in 1965 and it is now commonly referred to as the *Nelder-Mead simplex algorithm*. A contemporary view of the algorithm is provided in the well-written paper by Lagarias et al. [82]. In our exposition, we use the notation of this paper.

The Nelder-Mead algorithm is a derivative-free method. The method uses the concept of a simplex. A *simplex* is a geometric object determined by an assembly of $n + 1$ points, $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_n$, in the n -dimensional space such that

$$\det \begin{bmatrix} \mathbf{p}_0 & \mathbf{p}_1 & \cdots & \mathbf{p}_n \\ 1 & 1 & \cdots & 1 \end{bmatrix} \neq 0.$$

This condition ensures that two points in \mathbb{R} do not coincide, three points in \mathbb{R}^2 are not collinear, four points in \mathbb{R}^3 are not coplanar, and so on. Thus, simplex in \mathbb{R} is a line segment, in \mathbb{R}^2 it is a triangle, while a simplex in \mathbb{R}^3 is a tetrahedron; in each case it encloses a finite n -dimensional volume.

Suppose that we wish to minimize $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$. To start the algorithm, we initialize a simplex of $n + 1$ points. A possible way to set up a simplex, as suggested by Jang, Sun, and Mizutani [67], is to start with an initial point $\mathbf{x}^{(0)} = \mathbf{p}_0$ and generate the remaining points of the initial simplex as follows:

$$\mathbf{p}_i = \mathbf{p}_0 + \lambda_i \mathbf{e}_i, \quad i = 1, 2, \dots, n,$$

where the \mathbf{e}_i are unit vectors constituting the natural basis of \mathbb{R}^n as described in Section 2.1. The positive constant coefficients λ_i are selected in such a way that their magnitudes reflect the length scale of the optimization problem. Our objective is to modify the initial simplex stage by stage so that the resulting simplices converge toward the minimizer. At each iteration we evaluate the function f at each point of the simplex. In the function minimization process, the point with the largest function value is replaced with another point. The process for modifying the simplex continues until it converges toward the function minimizer.

We now present the rules for modifying the simplex stage by stage. To aid in our presentation, we use a two-dimensional example to illustrate the rules. We begin by selecting the initial set of $n + 1$ points that are to form the initial simplex. We next evaluate f at each point and order the $n + 1$ vertices to satisfy

$$f(\mathbf{p}_0) \leq f(\mathbf{p}_1) \leq \cdots \leq f(\mathbf{p}_n).$$

For the two-dimensional case we let \mathbf{p}_l , \mathbf{p}_{nl} , and \mathbf{p}_s denote the points of the simplex for which f is largest, next largest, and smallest; that is, because we wish to minimize f , the vertex \mathbf{p}_s is the best vertex, \mathbf{p}_l is the worst vertex, and \mathbf{p}_{nl} is the next-worst vertex. We next compute \mathbf{p}_g , the centroid (center

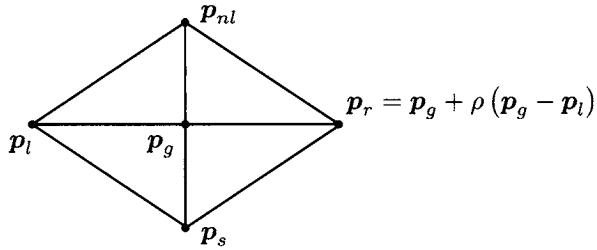


Figure 14.1 Reflecting \mathbf{p}_l in \mathbf{p}_g with a reflection coefficient ρ .

of gravity) of the best n points:

$$\mathbf{p}_g = \sum_{i=0}^{n-1} \frac{\mathbf{p}_i}{n}.$$

In our two-dimensional case, $n = 2$, we would have

$$\mathbf{p}_g = \frac{1}{2} (\mathbf{p}_{nl} + \mathbf{p}_s).$$

We then reflect the worst vertex, \mathbf{p}_l , in \mathbf{p}_g using a reflection coefficient $\rho > 0$ to obtain the reflection point

$$\mathbf{p}_r = \mathbf{p}_g + \rho (\mathbf{p}_g - \mathbf{p}_l).$$

A typical value is $\rho = 1$. The operation above is illustrated in Figure 14.1. We proceed to evaluate f at \mathbf{p}_r to obtain $f_r = f(\mathbf{p}_r)$. If $f_0 \leq f_r < f_{n-1}$ [i.e., if f_r lies between $f_s = f(\mathbf{p}_s)$ and $f_{nl} = f(\mathbf{p}_{nl})$], then the point \mathbf{p}_r replaces \mathbf{p}_l to form a new simplex, and we terminate the iteration. We proceed to repeat the process. Thus, we compute the centroid of the best n vertices of the new simplex and again reflect the point with the largest function f value in the centroid obtained for the best n points of the new simplex.

If, however, $f_r < f_s = f_0$, so that the point \mathbf{p}_r yields the smallest function value among the points of the simplex, we argue that this direction is a good one. In this case we increase the distance traveled using an *expansion coefficient* $\chi > 1$ (e.g., $\chi = 2$) to obtain

$$\mathbf{p}_e = \mathbf{p}_g + \chi (\mathbf{p}_r - \mathbf{p}_g).$$

The operation above yields a new point on the line $\mathbf{p}_l\mathbf{p}_g\mathbf{p}_r$ extended beyond \mathbf{p}_r . We illustrate this operation in Figure 14.2. If $f_e < f_r$ now, the expansion is declared a success and \mathbf{p}_e replaces \mathbf{p}_l in the next simplex. If, on the other hand, $f_e \geq f_r$, the expansion is a failure and \mathbf{p}_r replaces \mathbf{p}_l .

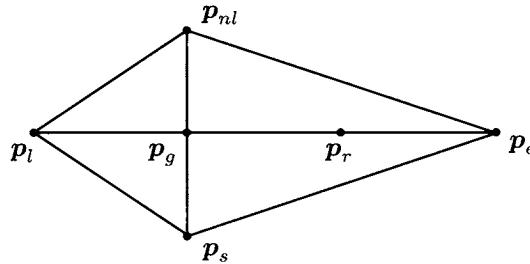


Figure 14.2 Expansion operation with the expansion coefficient χ .

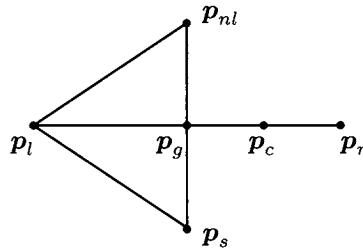


Figure 14.3 Outside contraction operation for the case when $f_r \in [f_{nl}, f_l)$.

Finally, if $f_r \geq f_{nl}$, the reflected point p_r would constitute the point with the largest function value in the new simplex. Then in the next step it would be reflected in p_g , probably an unfruitful operation. Instead, this case is dealt with by a *contraction* operation in one of two ways. First, if $f_r \geq f_{nl}$ and $f_r < f_l$, then we contract $(p_r - p_g)$ with a contraction coefficient $0 < \gamma < 1$ (e.g., $\gamma = 1/2$) to obtain

$$\mathbf{p}_c = \mathbf{p}_g + \gamma (\mathbf{p}_r - \mathbf{p}_g).$$

We refer to this operation as the *outside contraction*. See Figure 14.3 for an illustration of this operation. If, on the other hand, $f_r \geq f_{nl}$ and $f_r \geq f_l$, then p_l replaces p_r in the contraction operation and we get

$$\mathbf{p}_c = \mathbf{p}_g + \gamma (\mathbf{p}_l - \mathbf{p}_g).$$

This operation, referred to as the *inside contraction*, is illustrated in Figure 14.4.

If, in either case, $f_c \leq f_l$, the contraction is considered a success, and we replace p_l with p_c in the new simplex. If, however, $f_c > f_l$, the contraction

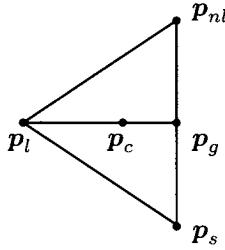


Figure 14.4 Inside contraction operation for the case when $f_r \geq f_l$.

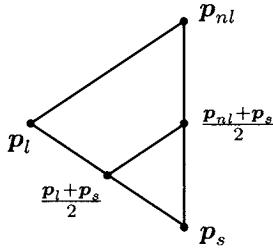


Figure 14.5 Shrinkage operation.

is a failure, and in this case a new simplex can be formed by retaining \mathbf{p}_s only and halving the distance from \mathbf{p}_s to every other point in the simplex. We can refer to this event as a shrinkage operation. The shrinkage operation is illustrated in Figure 14.5. In general, the shrink step produces the n new vertices of the new simplex according to the formula

$$\mathbf{v}_i = \mathbf{p}_s + \sigma(\mathbf{p}_i - \mathbf{p}_s), \quad i = 1, 2, \dots, n,$$

where $\sigma = 1/2$. Hence, the vertices of the new simplex are $\mathbf{p}_s, \mathbf{v}_1, \dots, \mathbf{v}_n$.

When implementing the simplex algorithm, we need a tie-breaking rule to order points in the case of equal function values. Lagarias et al. [82] propose tie-breaking rules that assign to the new vertex the highest possible index consistent with the relation

$$f(\mathbf{p}_0) \leq f(\mathbf{p}_1) \leq \cdots \leq f(\mathbf{p}_n).$$

In Figure 14.6 we illustrate the simplex search method by showing the first few stages of the search for a minimizer of a function of two variables. Our

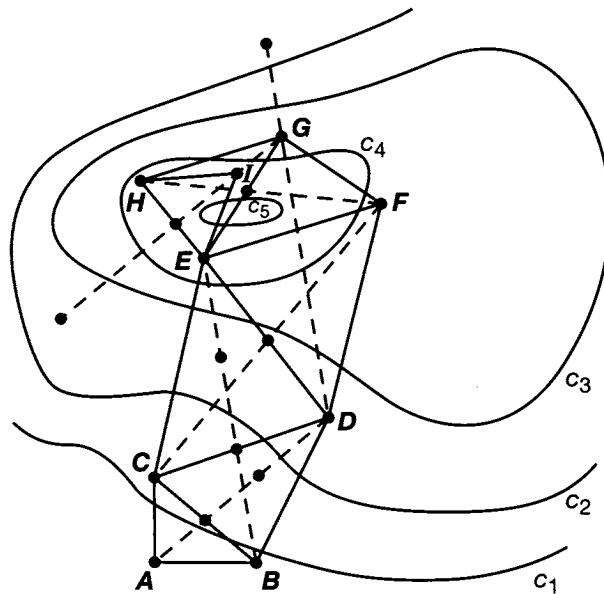


Figure 14.6 The simplex search method applied to minimization of a function of two variables.

drawing is inspired by a figure in Layton [84, p. 225]. The starting simplex is composed of the vertices A , B , and C . The vertices D and E are obtained by the expansion operation. The vertex F is obtained by the reflection operation. The vertex G is obtained using the outside contraction operation, while the vertex I is obtained employing the inside contraction operation. For the sake of clarity we terminate the process with the simplex composed of the vertices E , H , and I . The process may, of course, be continued beyond this simplex.

We add that a variant of the simplex method described above is presented in Jang et al. [67], where they use the centroid of the entire simplex rather than the centroid of the best n vertices of the simplex. That is, Jang et al. [67] compute the point p_g using the $n+1$ vertices of the simplex. In addition, they use only the inside contraction and they do not use the outside contraction operation.

14.3 Simulated Annealing

Randomized Search

Simulated annealing is an instance of a randomized search method. A *randomized search method*, also sometimes called a *probabilistic search method*, is an algorithm that searches the feasible set of an optimization problem by

considering randomized samples of candidate points in the set. The simulated annealing algorithm was first suggested for optimization by Kirkpatrick et al. [75] based on techniques of Metropolis et al. [91]. An early application to image processing was described by Geman and Geman [48].

As usual, suppose that we wish to solve an optimization problem of the form

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}) \\ & \text{subject to} && \mathbf{x} \in \Omega. \end{aligned}$$

The basic assumption in randomized search is our ability to select a random sample from the feasible set Ω . Typically, we start a randomized search process by selecting a random initial point $\mathbf{x}^{(0)} \in \Omega$. Then, we select a random next-candidate point, usually close to $\mathbf{x}^{(0)}$.

More formally, we assume that for any $\mathbf{x} \in \Omega$, there is a set $N(\mathbf{x}) \subset \Omega$ such that we can generate a random sample from this set. Typically, $N(\mathbf{x})$ is a set of points that are “close” to \mathbf{x} , and for this reason we usually think of $N(\mathbf{x})$ as a “neighborhood” of \mathbf{x} [we use the term *neighborhood* for $N(\mathbf{x})$ even in the general case where the points in it are arbitrary, not necessarily close to \mathbf{x}]. When we speak of generating a random point in $N(\mathbf{x})$, we mean that there is a prespecified distribution over $N(\mathbf{x})$, and we sample a point with this distribution. Often, this distribution is chosen to be uniform over $N(\mathbf{x})$; other distributions are also used, including Gaussian and Cauchy.

Before discussing the simulated annealing method, we first consider a simple randomized search algorithm, which we will call *naive random search*.

Naive Random Search Algorithm

1. Set $k := 0$. Select an initial point $\mathbf{x}^{(0)} \in \Omega$.
2. Pick a candidate point $\mathbf{z}^{(k)}$ at random from $N(\mathbf{x}^{(k)})$.
3. If $f(\mathbf{z}^{(k)}) < f(\mathbf{x}^{(k)})$, then set $\mathbf{x}^{(k+1)} = \mathbf{z}^{(k)}$; else, set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$.
4. If stopping criterion satisfied, then stop.
5. Set $k := k + 1$, go to step 2.

Note that the algorithm above has the familiar form $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$, where $\mathbf{d}^{(k)}$ is randomly generated. By design, the direction $\mathbf{d}^{(k)}$ either is $\mathbf{0}$ or is a descent direction. Typical stopping criteria include reaching a certain number of iterations, or reaching a certain objective function value.

Simulated Annealing Algorithm

The main problem with the naive random search method is that it may get stuck in a region around a local minimizer. This is easy to imagine; for

example, if $\mathbf{x}^{(0)}$ is a local minimizer and $N(\mathbf{x}^{(0)})$ is sufficiently small that all points in it have no smaller objective function value than $\mathbf{x}^{(0)}$ [i.e., $\mathbf{x}^{(0)}$ is a global minimizer of f over $N(\mathbf{x}^{(0)})$], then clearly the algorithm will be stuck and will never find a point outside of $N(\mathbf{x}^{(0)})$. To prevent getting stuck in a region around a local minimizer, we need a way to consider points outside this region. One way to achieve this goal is to make sure that at each k , the neighborhood $N(\mathbf{x}^{(k)})$ is a very large set. Indeed, if $N(\mathbf{x}^{(k)})$ is sufficiently large, then we are guaranteed that the algorithm will converge (in some sense) to a global minimizer. An extreme example of this case is where $N(\mathbf{x}) = \Omega$ for any $\mathbf{x} \in \Omega$ (in this case running k iterations of the naive random search algorithm amounts to finding the best point among k randomly chosen points in Ω). However, having too large a neighborhood in the search algorithm results in a slow search process, because the sampling of candidate points to consider is spread out, making it more unlikely to find a better candidate point.

Another way to overcome the problem of getting stuck in a region around a local minimizer is to modify the naive search algorithm so that we can “climb out” of such a region. This means that the algorithm may accept a new point that is *worse* than the current point. The simulated annealing algorithm incorporates such a mechanism.

Simulated Annealing Algorithm

1. Set $k := 0$; select an initial point $\mathbf{x}^{(0)} \in \Omega$.
2. Pick a candidate point $\mathbf{z}^{(k)}$ at random from $N(\mathbf{x}^{(k)})$.
3. Toss a coin with probability of HEAD equal to $p(k, f(\mathbf{z}^{(k)}), f(\mathbf{x}^{(k)}))$. If HEAD, then set $\mathbf{x}^{(k+1)} = \mathbf{z}^{(k)}$; else, set $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)}$.
4. If the stopping criterion is satisfied, then stop.
5. Set $k := k + 1$, go to step 2.

In step 3, the use of a “coin toss” is simply descriptive for a randomized decision—we do not mean literally that an actual coin needs to be tossed.

As in naive random search, the simulated annealing algorithm above has the familiar form $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$, where $\mathbf{d}^{(k)}$ is randomly generated. But in simulated annealing the direction $\mathbf{d}^{(k)}$ might be an ascent direction. However, as the algorithm progresses, we can keep track of the *best-so-far point*—this is a point $\mathbf{x}_{\text{best}}^{(k)}$ which, at each k , is equal to a $\mathbf{x}^{(j)}$, $j \in \{0, \dots, k\}$, such that $f(\mathbf{x}^{(j)}) \leq f(\mathbf{x}^{(i)})$ for all $i \in \{0, \dots, k\}$. The best-so-far point can be updated at each step k as follows:

$$\mathbf{x}_{\text{best}}^{(k)} = \begin{cases} \mathbf{x}^{(k)} & \text{if } f(\mathbf{x}^{(k)}) < f(\mathbf{x}_{\text{best}}^{(k-1)}) \\ \mathbf{x}_{\text{best}}^{(k-1)} & \text{otherwise.} \end{cases}$$

By keeping track of the best-so-far point, we can treat the simulated annealing algorithm simply as a search procedure; the best-so-far point is what we eventually use when the algorithm stops. This comment applies not only to simulated annealing, but other search techniques as well (including the randomized algorithms presented in the next two sections).

The major difference between simulated annealing and naive random search is that in step 5, there is some probability that we set the next iterate to be equal to the random point selected from the neighborhood, even if that point turns out to be worse than the current iterate. This probability is called the *acceptance probability*. For the algorithm to work properly, the acceptance probability must be chosen appropriately. A typical choice is

$$p(k, f(\mathbf{z}^{(k)}), f(\mathbf{x}^{(k)})) = \min\{1, \exp(-(f(\mathbf{z}^{(k)}) - f(\mathbf{x}^{(k)}))/T_k)\},$$

where \exp is the exponential function and T_k represents a positive sequence called the *temperature schedule* or *cooling schedule*. This form of acceptance probability is usually credited to Boltzmann and leads to a simulated annealing algorithm that behaves as a Gibbs sampler (a method of probabilistic sampling based on the Gibbs distribution).

Notice that if $f(\mathbf{z}^{(k)}) \leq f(\mathbf{x}^{(k)})$, then $p(k, f(\mathbf{z}^{(k)}), f(\mathbf{x}^{(k)})) = 1$, which means that we set $\mathbf{x}^{(k+1)} = \mathbf{z}^{(k)}$ (i.e., we move to the point $\mathbf{z}^{(k)}$). However, if $f(\mathbf{z}^{(k)}) > f(\mathbf{x}^{(k)})$, there is still a positive probability of setting $\mathbf{x}^{(k+1)} = \mathbf{z}^{(k)}$; this probability is equal to

$$\exp\left(-\frac{f(\mathbf{z}^{(k)}) - f(\mathbf{x}^{(k)})}{T_k}\right).$$

Note that the larger the difference between $f(\mathbf{z}^{(k)})$ and $f(\mathbf{x}^{(k)})$, the less likely we are to move to the worse point $\mathbf{z}^{(k)}$. Similarly, the smaller the value of T_k , the less likely we are to move to $\mathbf{z}^{(k)}$. It is typical to let the “temperature” T_k be monotonically decreasing to 0 (hence the word *cooling*). In other words, as the iteration index k increases, the algorithm becomes increasingly reluctant to move to a worse point. The intuitive reason for this behavior is that initially we wish to actively explore the feasible set, but with time we would like to be less active in exploration so that we spend more time in a region around a global minimizer. In other words, the desired behavior is this: Initially, the algorithm jumps around and is more likely to climb out of regions around local minimizers, but with time it settles down and is more likely to spend time around a global minimizer.

The term *annealing* comes from the field of metallurgy, where it refers to a technique for improving the property of metals. The basic procedure is to heat up a piece of metal and then cool it down in a controlled fashion. When the metal is first heated, the atoms in it become unstuck from their initial positions (with some level of internal energy). Then, as cooling takes place, the atoms gradually configure themselves in states of lower internal energy. Provided that the cooling is sufficiently slow, the final internal energy is lower

than the initial internal energy, thereby refining the crystalline structure and reducing defects.

In an analogous way, the temperature in simulated annealing must be cooled in a controlled fashion. In particular, the cooling should be sufficiently slow. In a seminal paper, Hajek [56] provides a rigorous analysis of the cooling schedule for convergence of the algorithm to a global minimizer. Specifically, he shows that an appropriate cooling schedule is

$$T_k = \frac{\gamma}{\log(k+2)},$$

where $\gamma > 0$ is a problem-dependent constant (large enough to allow the algorithm to “climb out” of regions around local minimizers that are not global minimizers). See also [57] for an analysis of a generalized version of simulated annealing.

Simulated annealing is often also used in combinatorial optimization, where the feasible set is finite (but typically large). An example of such a problem is the celebrated *traveling salesperson problem*. In the most basic form of this problem, we are given a number of cities and the cost of traveling from any city to any other city. The optimization problem is to find the cheapest round-trip route, starting from a given city, that visits every other city exactly once. For a description of how to apply simulated annealing to the traveling salesperson problem, see [67, p. 183].

14.4 Particle Swarm Optimization

Particle swarm optimization (PSO) is a randomized search technique presented by James Kennedy (a social psychologist) and Russell C. Eberhart (an engineer) in 1995 [73]. This optimization method is inspired by social interaction principles. The PSO algorithm differs from the randomized search methods discussed in Section 14.3 in one key way: Instead of updating a single candidate solution $\mathbf{x}^{(k)}$ at each iteration, we update a *population* (set) of candidate solutions, called a *swarm*. Each candidate solution in the swarm is called a *particle*. We think of a swarm as an apparently disorganized population of moving individuals that tend to cluster together while each individual seems to be moving in a random direction. (This description was adapted from a presentation by R. C. Eberhart.) The PSO algorithm aims to mimic the social behavior of animals and insects, such as a swarm of bees, a flock of birds, or a herd of wildebeest.

Suppose that we wish to minimize an objective function over \mathbb{R}^n . In the PSO algorithm, we start with an initial randomly generated population of points in \mathbb{R}^n . Associated with each point in the population is a velocity vector. We think of each point as the position of a particle, moving with an associated velocity. We then evaluate the objective function at each point in the population. Based on this evaluation, we create a new population of

points together with a new set of velocities. The creation of points in the new population, and their velocities, involve certain operations on points and velocities of the particles in the preceding population, described later.

Each particle keeps track of its *best-so-far position*—this is the best position it has visited so far (with respect to the value of the objective function). We will call this particle-dependent best-so-far position a *personal best* ($pbest$). In contrast, the overall best-so-far position (best among *all* the positions encountered so far by the entire population) is called a *global best* ($gbest$).

The particles “interact” with each other by updating their velocities according to their individual personal best as well as the global best. In the $gbest$ version of the PSO algorithm, presented below, the velocity of each particle is changed, at each time step, toward a combination of its $pbest$ and the $gbest$ locations. The velocity is weighted by a random term, with separate random numbers being generated for velocities toward $pbest$ and $gbest$ locations. Thus, the particles are drawn both to their own personal best positions as well as to the best position of the entire swarm. As usual, typical stopping criteria of the algorithm consist of reaching a certain number of iterations, or reaching a certain objective function value.

Basic PSO Algorithm

We now present a simple version of the $gbest$ version of the PSO algorithm, where at each time step the velocity of each particle is changed toward its $pbest$ and the $gbest$ locations. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be the objective function that we wish to minimize. Let d be the population size, and index the particles in the swarm by $i = 1, \dots, d$. Denote the position of particle i by $\mathbf{x}_i \in \mathbb{R}^n$ and its velocity by $\mathbf{v}_i \in \mathbb{R}^n$. Let \mathbf{p}_i be the $pbest$ of particle i and \mathbf{g} the $gbest$.

It is convenient to introduce the *Hadamard product* (or *Schur product*) operator, denoted by \circ : If \mathbf{A} and \mathbf{B} are matrices with the same dimension, then $\mathbf{A} \circ \mathbf{B}$ is a matrix of the same dimension as \mathbf{A} (or \mathbf{B}) resulting from entry-by-entry multiplication of \mathbf{A} and \mathbf{B} . This operation is denoted in MATLAB by “ $.*$ ” (the dot before an operator indicates entry-by-entry operations). Thus, if \mathbf{A} and \mathbf{B} have the same dimension, then $\mathbf{A}.*\mathbf{B}$ returns a matrix whose entries are simply the products of the corresponding individual entries of \mathbf{A} and \mathbf{B} . The PSO $gbest$ algorithm uses three given constant real parameters, ω , c_1 , and c_2 , which we discuss after presenting the algorithm.

PSO Gbest Algorithm

- Set $k := 0$. For $i = 1, \dots, d$, generate initial random positions $\mathbf{x}_i^{(0)}$ and velocities $\mathbf{v}_i^{(0)}$, and set $\mathbf{p}_i^{(0)} = \mathbf{x}_i^{(0)}$. Set $\mathbf{g}^{(0)} = \arg \min_{\mathbf{x} \in \{\mathbf{x}_1^{(0)}, \dots, \mathbf{x}_d^{(0)}\}} f(\mathbf{x})$.

2. For $i = 1, \dots, d$, generate random n -vectors $\mathbf{r}_i^{(k)}$ and $\mathbf{s}_i^{(k)}$ with components uniformly in the interval $(0, 1)$, and set

$$\begin{aligned}\mathbf{v}_i^{(k+1)} &= \omega \mathbf{v}_i^{(k)} + c_1 \mathbf{r}_i^{(k)} \circ (\mathbf{p}_i^{(k)} - \mathbf{x}_i^{(k)}) + c_2 \mathbf{s}_i^{(k)} \circ (\mathbf{g}^{(k)} - \mathbf{x}_i^{(k)}), \\ \mathbf{x}_i^{(k+1)} &= \mathbf{x}_i^{(k)} + \mathbf{v}_i^{(k+1)}.\end{aligned}$$
3. For $i = 1, \dots, d$, if $f(\mathbf{x}_i^{(k+1)}) < f(\mathbf{p}_i^{(k)})$, then set $\mathbf{p}_i^{(k+1)} = \mathbf{x}_i^{(k+1)}$; else, set $\mathbf{p}_i^{(k+1)} = \mathbf{p}_i^{(k)}$.
4. If there exists $i \in \{1, \dots, d\}$ such that $f(\mathbf{x}_i^{(k+1)}) < f(\mathbf{g}^{(k)})$, then set $\mathbf{g}^{(k+1)} = \mathbf{x}_i^{(k+1)}$; else, set $\mathbf{g}^{(k+1)} = \mathbf{g}^{(k)}$.
5. If stopping criterion satisfied, then stop.
6. Set $k := k + 1$, go to step 2.

In the algorithm, the parameter ω is referred to as an *inertial constant*. Recommended values are slightly less than 1. The parameters c_1 and c_2 are constants that determine how much the particle is directed toward “good” positions. They represent a “cognitive” and a “social” component, respectively, in that they affect how much the particle’s personal best and the global best influence its movement. Recommended values are $c_1, c_2 \approx 2$.

Variations

The PSO techniques have evolved since 1995. For example, recently Clerc [29] proposed a *constriction-factor* version of the algorithm, where the velocity is updated as

$$\mathbf{v}_i^{(k+1)} = \kappa \left(\mathbf{v}_i^{(k)} + c_1 \mathbf{r}_i^{(k)} \circ (\mathbf{p}_i^{(k)} - \mathbf{x}_i^{(k)}) + c_2 \mathbf{s}_i^{(k)} \circ (\mathbf{g}^{(k)} - \mathbf{x}_i^{(k)}) \right),$$

where the *constriction coefficient* κ is computed as

$$\kappa = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|},$$

where $\phi = c_1 + c_2$ and $\phi > 4$. For example, for $\phi = 4.1$, we have $\kappa = 0.729$. The role of the constriction coefficient is to speed up the convergence.

When using PSO in practice, one might wish to clamp the velocities to a certain maximum amount, say, v_{\max} . In other words, we replace each component v of the velocity vector by

$$\min \{v_{\max}, \max\{-v_{\max}, v\}\}.$$

For an up-to-date literature survey and other modifications and heuristics, we recommend the first part of the proceedings of the *8th International Conference on Adaptive and Natural Computing Algorithms*, held in April 2007 in

Warsaw, Poland [5]. In these proceedings, one can find a number of papers dealing with applications of PSO to multiobjective optimization problems, versions of PSO for constrained optimization problems, as well as “niching” versions designed to find multiple solutions, that is, applications of PSO to multimodal optimization problems. For a mathematical analysis of the PSO algorithm, see Clerc and Kennedy [30].

14.5 Genetic Algorithms

Basic Description

A *genetic algorithm* is a randomized, population-based search technique that has its roots in the principles of genetics. The beginnings of genetic algorithms is credited to John Holland, who developed the basic ideas in the late 1960s and early 1970s. Since its conception, genetic algorithms have been used widely as a tool in computer programming and artificial intelligence (e.g., [61], [79], and [94]), optimization (e.g., [36], [67], and [127]), neural network training (e.g., [80]), and many other areas.

Suppose that we wish to solve an optimization problem of the form

$$\begin{aligned} & \text{maximize } f(\mathbf{x}) \\ & \text{subject to } \mathbf{x} \in \Omega \end{aligned}$$

(notice that the problem is a maximization, which is more convenient for describing genetic algorithms). The underlying idea of genetic algorithms applied to this problem is as follows. We start with an initial set of points in Ω , denoted $P(0)$, called the *initial population*. We then evaluate the objective function at points in $P(0)$. Based on this evaluation, we create a new set of points $P(1)$. The creation of $P(1)$ involves certain operations on points in $P(0)$, called *crossover* and *mutation*, discussed later. We repeat the procedure iteratively, generating populations $P(2), P(3), \dots$, until an appropriate stopping criterion is reached. The purpose of the crossover and mutation operations is to create a new population with an average objective function value that is higher than that of the previous population. To summarize, the genetic algorithm iteratively performs the operations of crossover and mutation on each population to produce a new population until a chosen stopping criterion is met.

The terminology used in describing genetic algorithms is adopted from genetics. To proceed with describing the details of the algorithm, we need the additional ideas and terms described below.

Chromosomes and Representation Schemes First, we point out that, in fact, genetic algorithms do not work directly with points in the set Ω , but rather, with an *encoding* of the points in Ω . Specifically, we need first to map Ω onto a set consisting of strings of symbols, all of equal length. These strings are

called *chromosomes*. Each chromosome consists of elements from a chosen set of symbols, called the *alphabet*. For example, a common alphabet is the set $\{0, 1\}$, in which case the chromosomes are simply binary strings. We denote by L the length of chromosomes (i.e., the number of symbols in the strings). To each chromosome there corresponds a value of the objective function, referred to as the *fitness* of the chromosome. For each chromosome \mathbf{x} , we write $f(\mathbf{x})$ for its fitness. Note that, for convenience, we use f to denote both the original objective function and the fitness measure on the set of chromosomes. We assume that f is a nonnegative function.

The choice of chromosome length, alphabet, and encoding (i.e., the mapping from Ω onto the set of chromosomes) is called the *representation scheme* for the problem. Identification of an appropriate representation scheme is the first step in using genetic algorithms to solve a given optimization problem.

Once a suitable representation scheme has been chosen, the next phase is to initialize the first population $P(0)$ of chromosomes. This is usually done by a random selection of a set of chromosomes. After we form the initial population of chromosomes, we then apply the operations of crossover and mutation on the population. During each iteration k of the process, we evaluate the fitness $f(\mathbf{x}^{(k)})$ of each member $\mathbf{x}^{(k)}$ of the population $P(k)$. After the fitness of the entire population has been evaluated, we form a new population $P(k+1)$ in two stages.

Selection and Evolution In the first stage we apply an operation called *selection*, where we form a set $M(k)$ with the same number of elements as $P(k)$. This number is called the *population size*, which we denote by N . The set $M(k)$, called the *mating pool*, is formed from $P(k)$ using a random procedure as follows: Each point $\mathbf{m}^{(k)}$ in $M(k)$ is equal to $\mathbf{x}^{(k)}$ in $P(k)$ with probability

$$\frac{f(\mathbf{x}^{(k)})}{F(k)},$$

where

$$F(k) = \sum f(\mathbf{x}_i^{(k)})$$

and the sum is taken over the whole of $P(k)$. In other words, we select chromosomes into the mating pool with probabilities proportional to their fitness.

The selection scheme above is also called the *roulette-wheel* scheme, for the following reason. Imagine a roulette wheel in which each slot is assigned to a chromosome in $P(k)$; some chromosomes may be assigned multiple slots. The number of slots associated with each chromosome is in proportion to its fitness. We then spin the roulette wheel and select [for inclusion in $M(k)$] the chromosome on whose slot the ball comes to rest. This procedure is repeated N times, so that the mating pool $M(k)$ contains N chromosomes.

An alternative selection scheme is the *tournament scheme*, which proceeds as follows. First, we select a pair of chromosomes at random from $P(k)$. We

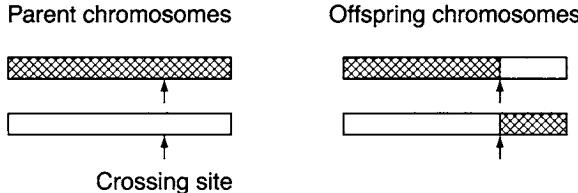


Figure 14.7 Illustration of basic crossover operation.

then compare the fitness values of these two chromosomes, and place the fitter of the two into $M(k)$. We repeat this operation until the mating pool $M(k)$ contains N chromosomes.

The second stage is called *evolution*: in this stage, we apply the crossover and mutation operations. The *crossover operation* takes a pair of chromosomes, called the *parents*, and gives a pair of *offspring chromosomes*. The operation involves exchanging substrings of the two parent chromosomes, described below. Pairs of parents for crossover are chosen from the mating pool randomly, such that the probability that a chromosome is chosen for crossover is p_c . We assume that whether or not a given chromosome is chosen is independent of whether or not any other chromosome is chosen for crossover.

We can pick parents for crossover in several ways. For example, we may randomly choose two chromosomes from the mating pool as parents. In this case if N is the number of chromosomes in the mating pool, then $p_c = 2/N$. Similarly, if we randomly pick $2k$ chromosomes from the mating pool (where $k < N/2$), forming k pairs of parents, we have $p_c = 2k/N$. In the two examples above, the number of pairs of parents is fixed and the value of p_c is dependent on this number. Yet another way of choosing parents is as follows: Given a value of p_c , we pick a random number of pairs of parents such that the average number of pairs is $p_cN/2$.

Once the parents for crossover have been determined, we apply the crossover operation to the parents. There are many types of possible crossover operations. The simplest crossover operation is the *one-point crossover*. In this operation, we first choose a number randomly between 1 and $L - 1$ according to a uniform distribution, where L is the length of chromosomes. We refer to this number as the *crossing site*. Crossover then involves exchanging substrings of the parents to the left of the crossing site, as illustrated in Figure 14.7 and in the following example.

Example 14.1 Suppose that we have chromosomes of length $L = 6$ over the binary alphabet $\{0, 1\}$. Consider the pair of parents 000000 and 111111. Suppose that the crossing site is 4. Then, the crossover operation applied to the parent chromosomes yields the two offspring 000011 and 111100. ■

We can also have crossover operations with multiple crossing sites, as illustrated in Figure 14.8 and in the following example.

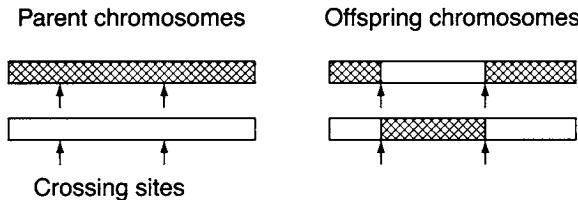


Figure 14.8 Illustration of two-point crossover operation.

Example 14.2 Consider two chromosomes, 000000000 and 111111111, of length $L = 9$. Suppose that we have two crossing sites, at 3 and 7. Then, the crossover operation applied to the parent chromosomes above yields the two offspring 000111100 and 111000011. ■

After the crossover operation, we replace the parents in the mating pool by their offspring. The mating pool has therefore been modified but maintains the same number of elements.

Next, we apply the *mutation operation*, which takes each chromosome from the mating pool and randomly changes each symbol of the chromosome with a given probability p_m . In the case of the binary alphabet, this change corresponds to complementing the corresponding bits; that is, we replace each bit with probability p_m from 0 to 1, or vice versa. If the alphabet contains more than two symbols, then the change involves randomly substituting the symbol with another symbol from the alphabet. Typically, the value of p_m is very small (e.g., 0.01), so that only a few chromosomes will undergo a change due to mutation, and of those that are affected, only a few of the symbols are modified. Therefore, the mutation operation plays only a minor role in the genetic algorithm relative to the crossover operation.

After applying the crossover and mutation operations to the mating pool $M(k)$, we obtain the new population $P(k+1)$. We then repeat the procedure of evaluation, selection, and evolution, iteratively. We summarize the genetic algorithm as follows.

Genetic Algorithm

1. Set $k := 0$. Generate an initial population $P(0)$.
2. Evaluate $P(k)$.
3. If the stopping criterion is satisfied, then stop.
4. Select $M(k)$ from $P(k)$.
5. Evolve $M(k)$ to form $P(k + 1)$.
6. Set $k := k + 1$, go to step 2.

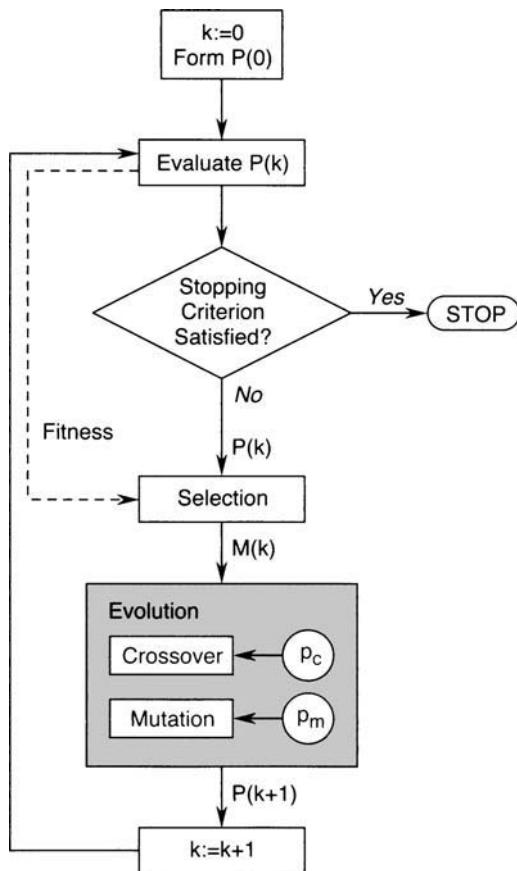


Figure 14.9 Flowchart for the genetic algorithm.

A flowchart illustrating this algorithm is shown in Figure 14.9.

During execution of the genetic algorithm, we keep track of the *best-so-far* chromosome, that is, the chromosome with the highest fitness of all the chromosomes evaluated. After each iteration, the best-so-far chromosome serves as the candidate for the solution to the original problem. Indeed, we may even copy the best-so-far chromosome into each new population, a practice referred to as *elitism*. The elitist strategy may result in domination of the population by “superchromosomes.” However, practical experience suggests that elitism often improves the performance of the algorithm.

The stopping criterion can be implemented in a number of ways. For example, a simple stopping criterion is to stop after a prespecified number of iterations. Another possible criterion is to stop when the fitness for the best-so-far chromosome does not change significantly from iteration to iteration.

The genetic algorithm differs from the algorithms discussed in previous chapters in several respects. First, it does not use derivatives of the objective function (like the other methods in this chapter). Second, it uses operations that are random within each iteration (like the other randomized search methods). Third, it searches from a set of points rather than a single point at each iteration (like the PSO algorithm). Fourth, it works with an encoding of the feasible set rather with than the set itself.

We illustrate an application of the genetic algorithm to an optimization problem in the following example.

Example 14.3 Consider the MATLAB “peaks” function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ given by

$$f(x, y) = 3(1 - x)^2 e^{-x^2 - (y+1)^2} - 10 \left(\frac{x}{5} - x^3 - y^5 \right) e^{-x^2 - y^2} - \frac{e^{-(x+1)^2 - y^2}}{3}$$

(see also [67, pp. 178–180] for an example involving the same function). We wish to maximize f over the set $\Omega = \{[x, y]^\top \in \mathbb{R}^2 : -3 \leq x, y \leq 3\}$. A plot of the objective function f over the feasible set Ω is shown in Figure 14.10. Using the MATLAB function `fminunc` (from the Optimization Toolbox), we found the optimal point to be $[-0.0093, 1.5814]^\top$, with objective function value 8.1062.

To apply the genetic algorithm to solve the optimization problem above, we use a simple binary representation scheme with length $L = 32$, where the first 16 bits of each chromosome encode the x component, whereas the remaining 16 bits encode the y component. Recall that x and y take values in the interval $[-3, 3]$. We first map the interval $[-3, 3]$ onto the interval $[0, 2^{16} - 1]$, via a simple translation and scaling. The integers in the interval $[0, 2^{16} - 1]$ are then expressed as binary 16-bit strings. This defines the encoding of each component x and y . The chromosome is obtained by juxtaposing the two 8-bit strings. For example, the point $[x, y]^\top = [-1, 3]^\top$ is encoded as (see

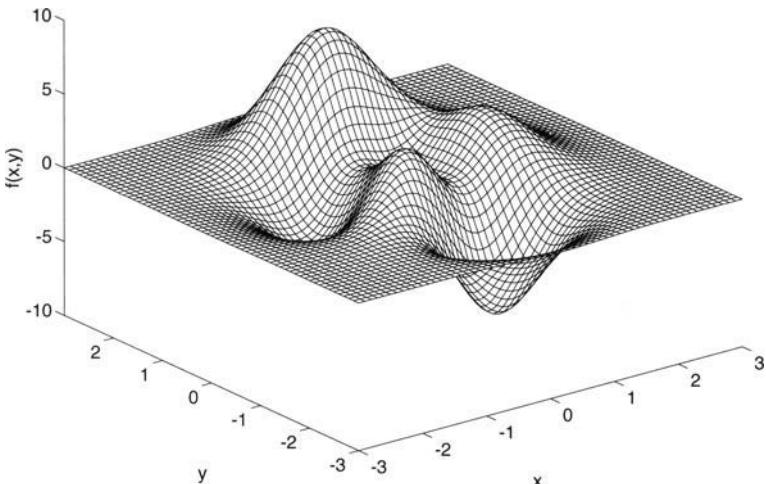


Figure 14.10 Plot of f for Example 14.3.

Exercise 14.4 for a simple algorithm for converting from decimal into binary)

$$\underbrace{01010101010101}_{\text{encoded } x = -1} \underbrace{1111111111111111}_{\text{encoded } y = 3}.$$

Using a population size of 20, we apply 50 iterations of the genetic algorithm on the problem above. We used parameter values of $p_c = 0.75$ and $p_m = 0.0075$. Figure 14.11 shows plots of the best, average, and worst objective function values in the population for every iteration (generation) of the algorithm. The best-so-far solution obtained at the end of the 50 iterations is $[0.0615, 1.5827]^\top$, with objective function value 8.1013. Note that this solution and objective function value are very close to those obtained using MATLAB. ■

Analysis of Genetic Algorithms

In this section we use heuristic arguments to describe why genetic algorithms work. As pointed out before, the genetic algorithm was motivated by ideas from natural genetics [61]. Specifically, the notion of “survival of the fittest” plays a central role. The mechanisms used in the genetic algorithm mimic this principle. We start with a population of chromosomes, and selectively pick the fittest ones for reproduction. From these selected chromosomes, we form the new generation by combining information encoded in them. In this way,

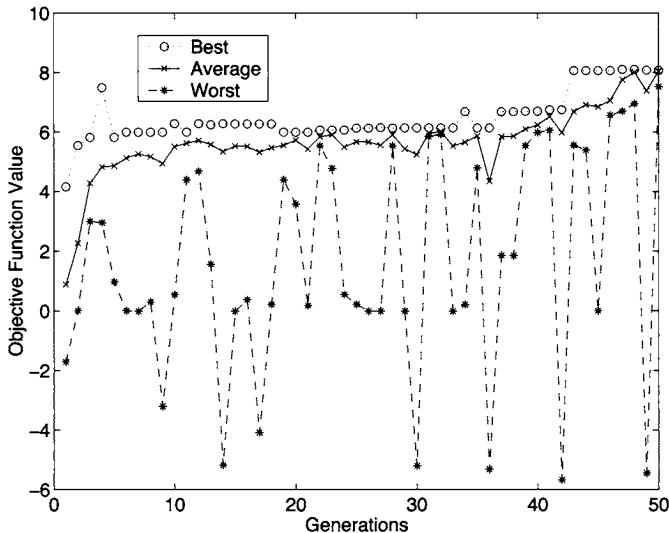


Figure 14.11 The best, average, and worst objective function values in the population for every iteration (generation) of the genetic algorithm in Example 14.3.

the goal is to ensure that the fittest members of the population survive and their information content is preserved and combined to produce even better offspring.

To further analyze the genetic algorithm in a more quantitative fashion, we need to define a few terms. For convenience, we only consider chromosomes over the binary alphabet. We introduce the notion of a *schema* (plural: *schemata*) as a set of chromosomes with certain common features. Specifically, a schema is a set of chromosomes that contain 1s and 0s in particular locations. We represent a schema using a string notation over an extended alphabet $\{0, 1, *\}$. For example, the notation $1 * 01$ represents the schema

$$1 * 01 = \{1001, 1101\},$$

and the notation $0 * 101*$ represents the schema

$$0 * 101* = \{001010, 001011, 011010, 011011\}.$$

In the schema notation, the numbers 0 and 1 denote the fixed binary values in the chromosomes that belong to the schema. The symbol *, meaning “don’t care,” matches either 0 or 1 at the positions it occupies. Thus, a schema describes a set of chromosomes that have certain specified similarities. A chromosome belongs to a particular schema if for all positions $j = 1, \dots, L$ the symbol found in the j th position of the chromosome matches the symbol found in the j th position of the schema, with the understanding that any symbol matches *. Note that if a schema has r “don’t care” symbols, then it

contains 2^r chromosomes. Moreover, any chromosome of length L belongs to 2^L schemata.

Given a schema that represents good solutions to our optimization problem, we would like the number of matching chromosomes in the population $P(k)$ to grow as k increases. This growth is affected by several factors, which we analyze in the following discussion. We assume throughout that we are using the roulette-wheel selection method.

The first key idea in explaining why the genetic algorithm works is the observation that if a schema has chromosomes with better-than-average fitness, then the expected (mean) number of chromosomes matching this schema in the mating pool $M(k)$ is larger than the number of chromosomes matching this schema in the population $P(k)$. To quantify this assertion, we need some additional notation. Let H be a given schema, and let $e(H, k)$ be the number of chromosomes in $P(k)$ that match H ; that is, $e(H, k)$ is the number of elements in the set $P(k) \cap H$. Let $f(H, k)$ be the average fitness of chromosomes in $P(k)$ that match schema H . This means that if $H \cap P(k) = \{\mathbf{x}_1, \dots, \mathbf{x}_{e(H, k)}\}$, then

$$f(H, k) = \frac{f(\mathbf{x}_1) + \dots + f(\mathbf{x}_{e(H, k)})}{e(H, k)}.$$

Let N be the number of chromosomes in the population and $F(k)$ be the sum of the fitness values of chromosomes in $P(k)$, as before. Denote by $\bar{F}(k)$ the average fitness of chromosomes in the population; that is,

$$\begin{aligned}\bar{F}(k) &= \frac{F(k)}{N} \\ &= \frac{1}{N} \sum f(\mathbf{x}_i^{(k)}).\end{aligned}$$

Finally, let $m(H, k)$ be the number of chromosomes in $M(k)$ that match H , in other words, the number of elements in the set $M(k) \cap H$.

Lemma 14.1 *Let H be a given schema and $\mathcal{M}(H, k)$ be the expected value of $m(H, k)$ given $P(k)$. Then,*

$$\mathcal{M}(H, k) = \frac{f(H, k)}{\bar{F}(k)} e(H, k).$$

□

Proof. Let $P(k) \cap H = \{\mathbf{x}_1, \dots, \mathbf{x}_{e(H, k)}\}$. In the remainder of the proof, the term *expected* should be taken to mean “expected, given $P(k)$.” For each element $\mathbf{m}^{(k)} \in M(k)$ and each $i = 1, \dots, e(H, k)$, the probability that $\mathbf{m}^{(k)} = \mathbf{x}_i$ is given by $f(\mathbf{x}_i)/F(k)$. Thus, the expected number of chromosomes in $M(k)$ equal to \mathbf{x}_i is

$$N \frac{f(\mathbf{x}_i)}{F(k)} = \frac{f(\mathbf{x}_i)}{\bar{F}(k)}.$$

Hence, the expected number of chromosomes in $P(k) \cap H$ that are selected into $M(k)$ is

$$\sum_{i=1}^{e(H,k)} \frac{f(\mathbf{x}_i)}{\bar{F}(k)} = e(H,k) \frac{\sum_{i=1}^{e(H,k)} f(\mathbf{x}_i)}{e(H,k)} \frac{1}{\bar{F}(k)} = \frac{f(H,k)}{\bar{F}(k)} e(H,k).$$

Because any chromosome in $M(k)$ is also a chromosome in $P(k)$, the chromosomes in $M(k) \cap H$ are simply those in $P(k) \cap H$ that are selected into $M(k)$. Hence,

$$\mathcal{M}(H, k) = \frac{f(H, k)}{\bar{F}(k)} e(H, k).$$

■

Lemma 14.1 quantifies our assertion that if a schema H has chromosomes with better than average fitness [i.e., $f(H, k)/\bar{F}(k) > 1$], then the expected number of chromosomes matching H in the mating pool $M(k)$ is larger than the number of chromosomes matching H in the population $P(k)$.

We now analyze the effect of the evolution operations on the chromosomes in the mating pool. For this, we need to introduce two parameters that are useful in the characterization of a schema: order and length. The *order* $o(S)$ of a schema S is the number of fixed symbols (non* symbols) in its representation (the notation $o(S)$ is standard in the literature on genetic algorithms, and should not be confused with the “little-oh” symbol defined in Section 5.6). If the *length* of chromosomes in S is L , then $o(S)$ is L minus the number of * symbols in S . For example,

$$o(1 * 01) = 4 - 1 = 3,$$

whereas

$$o(0 * 1 * 01) = 6 - 2 = 4.$$

The *length* $l(S)$ of a schema S is the distance between the first and last fixed symbols (i.e., the difference between the positions of the rightmost fixed symbol and the leftmost fixed symbol). For example,

$$l(1 * 01) = 4 - 1 = 3,$$

$$l(0 * 101*) = 5 - 1 = 4,$$

$$l(**1*) = 0.$$

Note that for a schema S with chromosomes of length L , the order $o(S)$ is a number between 0 and L and the length $l(S)$ is a number between 0 in $L - 1$. The order of a schema with all * symbols is 0; its length is also 0. The order of a schema containing only a single element (i.e., its representation has no * symbols) is L [e.g., $o(1011) = 4 - 0 = 4$]. The length of a schema with fixed symbols in its first and last positions is $L - 1$ [e.g., $l(0 ** 1) = 4 - 1 = 3$].

We first consider the effect of the crossover operation on the mating pool. The basic observation in the following lemma is that given a chromosome in $M(k) \cap H$, the probability that it leaves H after crossover is bounded above by a quantity that is proportional to p_c and $l(H)$.

Lemma 14.2 *Given a chromosome in $M(k) \cap H$, the probability that it is chosen for crossover and neither of its offspring is in H is bounded above by*

$$p_c \frac{l(H)}{L - 1}.$$

□

Proof. Consider a given chromosome in $M(k) \cap H$. The probability that it is chosen for crossover is p_c . If neither of its offspring is in H , then the crossover point must be between the corresponding first and last fixed symbols of H . The probability of this is $l(H)/(L - 1)$. Hence, the probability that the given chromosome is chosen for crossover and neither of its offspring is in H is bounded above by

$$p_c \frac{l(H)}{L - 1}.$$

■

From Lemma 14.2 we conclude that given a chromosome in $M(k) \cap H$, the probability either that it is not selected for crossover or that at least one of its offspring is in H after the crossover operation, is bounded below by

$$1 - p_c \frac{l(H)}{L - 1}.$$

Note that if a chromosome in H is chosen for crossover and the other parent chromosome is also in H , then both offspring are automatically in H (see Exercise 14.5). Hence, for each chromosome in $M(k) \cap H$, there is a certain probability that it will result in an associated chromosome in H (either itself or one of its offspring) after going through crossover (including selection for crossover) and that probability is bounded below by the foregoing expression.

We next consider the effect of the mutation operation on the mating pool $M(k)$.

Lemma 14.3 *Given a chromosome in $M(k) \cap H$, the probability that it remains in H after the mutation operation is given by*

$$(1 - p_m)^{o(H)}.$$

□

Proof. Given a chromosome in $M(k) \cap H$, it remains in H after the mutation operation if and only if none of the symbols in this chromosome that correspond to fixed symbols in H are changed by the mutation operation. The probability of this event is $(1 - p_m)^{o(H)}$. ■

Note that if p_m is small, the expression $(1 - p_m)^{o(H)}$ above is approximately equal to

$$1 - p_m o(H).$$

The following theorem combines the results of the preceding lemmas.

Theorem 14.1 *Let H be a given schema and $\mathcal{E}(H, k + 1)$ be the expected value of $e(H, k + 1)$ given $P(k)$. Then,*

$$\mathcal{E}(H, k + 1) \geq \left(1 - p_c \frac{l(H)}{L - 1}\right) (1 - p_m)^{o(H)} \frac{f(H, k)}{\bar{F}(k)} e(H, k).$$

□

Proof. Consider a given chromosome in $M(k) \cap H$. If, after the evolution operations, it has a resulting chromosome that is in H , then that chromosome is in $P(k + 1) \cap H$. By Lemmas 14.2 and 14.3, the probability of this event is bounded below by

$$\left(1 - p_c \frac{l(H)}{L - 1}\right) (1 - p_m)^{o(H)}.$$

Therefore, because each chromosome in $M(k) \cap H$ results in a chromosome in $P(k + 1) \cap H$ with a probability bounded below by the expression above, the expected value of $e(H, k + 1)$ given $M(k)$ is bounded below by

$$\left(1 - p_c \frac{l(H)}{L - 1}\right) (1 - p_m)^{o(H)} m(H, k).$$

Taking the expectation given $P(k)$, we get

$$\mathcal{E}(H, k + 1) \geq \left(1 - p_c \frac{l(H)}{L - 1}\right) (1 - p_m)^{o(H)} \mathcal{M}(H, k).$$

Finally, using Lemma 14.1, we arrive at the desired result. ■

Theorem 14.1 indicates how the number of chromosomes in a given schema changes from one population to the next. Three factors influence this change, reflected by the three terms on the right-hand side of inequality in Theorem 14.1: $1 - p_c l(H)/(L - 1)$, $(1 - p_m)^{o(H)}$, and $f(H, k)/\bar{F}(k)$. Note that the larger the values of these terms, the higher the expected number of matches of the schema H in the next population. The effect of each term is summarized as follows:

- The term $f(H, k)/\bar{F}(k)$ reflects the role of average fitness of the given schema H —the higher the average fitness, the higher the expected number of matches in the next population.
- The term $1 - p_c l(H)/(L - 1)$ reflects the effect of crossover—the smaller the term $p_c l(H)/(L - 1)$, the higher the expected number of matches in the next population.

- The term $(1-p_m)^{o(H)}$ reflects the effect of mutation—the larger the term, the higher the expected number of matches in the next population.

In summary, we see that a schema that is short, low order, and has above-average fitness will have on average an increasing number of its representatives in the population from iteration to iteration. Observe that the encoding is relevant to the performance of the algorithm. Specifically, a good encoding is one that results in high-fitness schemata having small lengths and orders.

Real-Number Genetic Algorithms

The genetic algorithms described thus far operate on binary strings, representing elements of the feasible set Ω . (For this reason, genetic algorithms are also suitably applied to combinatorial optimization problems, where Ω is not \mathbb{R}^n but some discrete set.) Binary encodings allow us to use the schema theory, described in the preceding section, to analyze genetic algorithms. However, there are some disadvantages to operating on binary strings. To see this, let $\mathbf{g} : \{0, 1\}^L \rightarrow \Omega$ represent the binary “decoding” function; that is, if \mathbf{x} is a binary chromosome, $\mathbf{g}(\mathbf{x}) \in \Omega$ is the point in the feasible set $\Omega \subset \mathbb{R}^n$ whose encoding is \mathbf{x} . Therefore, the objective function being maximized by the genetic algorithm is not f itself but rather the composition of f and the decoding function \mathbf{g} . In other words, the optimization problem being solved by the genetic algorithm is

$$\begin{aligned} &\text{maximize} && f(\mathbf{g}(\mathbf{x})) \\ &\text{subject to} && \mathbf{x} \in \{\mathbf{y} \in \{0, 1\}^L : \mathbf{g}(\mathbf{y}) \in \Omega\}. \end{aligned}$$

This optimization problem may be more complex than the original optimization problem. For example, it may have extra maximizers, making the search for a global maximizer more difficult.

The above motivates a consideration of genetic algorithms that operate directly on the original optimization problem. In other words, we wish to implement a genetic algorithm that operates directly on \mathbb{R}^n . The steps of this algorithm will be the same as before (see Figure 14.9), except that the elements of the population are points in the feasible set Ω rather than binary strings. We will need to define appropriate crossover and mutation operations for this case.

For crossover, we have several options. The simplest is to use averaging: For a pair of parents \mathbf{x} and \mathbf{y} , the offspring is $\mathbf{z} = (\mathbf{x} + \mathbf{y})/2$ (this type of crossover operation is used, e.g., in [103]). This offspring can then replace one of the parents. Alternatively, we may produce two offspring as follows: $\mathbf{z}_1 = (\mathbf{x} + \mathbf{y})/2 + \mathbf{w}_1$ and $\mathbf{z}_2 = (\mathbf{x} + \mathbf{y})/2 + \mathbf{w}_2$, where \mathbf{w}_1 and \mathbf{w}_2 are two randomly generated vectors (with zero mean). If either offspring lies outside Ω , we have to bring the offspring back into Ω , using, for example, a projection (see Section 23.2). A third option for crossover is to take random convex combinations of the parents. Specifically, given a pair of parents \mathbf{x} and \mathbf{y} ,

we generate a random number $\alpha \in (0, 1)$ and then produce two offspring $\mathbf{z}_1 = \alpha\mathbf{x} + (1 - \alpha)\mathbf{y}$ and $\mathbf{z}_2 = (1 - \alpha)\mathbf{x} + \alpha\mathbf{y}$. This method of crossover ensures that the offspring are always in the feasible set, provided that the feasible set is convex. A fourth option is to perturb the two points above by some random amount: $\mathbf{z}_1 = \alpha\mathbf{x} + (1 - \alpha)\mathbf{y} + \mathbf{w}_1$ and $\mathbf{z}_2 = (1 - \alpha)\mathbf{x} + \alpha\mathbf{y} + \mathbf{w}_2$, where \mathbf{w}_1 and \mathbf{w}_2 are two randomly generated vectors (with zero mean). In this case we have to check for feasibility of the offspring and use projections if needed.

For mutation, a simple implementation is to add a random vector to the chromosome. Specifically, given a chromosome \mathbf{x} , we produce its mutation as $\mathbf{x}' = \mathbf{x} + \mathbf{w}$, where \mathbf{w} is a random vector with zero mean. This mutation operation is also called a *real number creep* (see, e.g., [103]). As before, we have to ensure that the mutated chromosome is feasible. If not, we may use a projection. An alternative method for mutation is to replace the chosen chromosome with a random convex combination of the chromosome with a random point in the feasible set; that is, we generate a random number $\alpha \in (0, 1)$ and a random point $\mathbf{w} \in \Omega$, and set $\mathbf{x}' = \alpha\mathbf{x} + (1 - \alpha)\mathbf{w}$. Provided that the feasible set is convex, the mutated chromosome will always be feasible.

Example 14.4 Consider again the function $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ from Example 14.3. We apply a real-number genetic algorithm to find a maximizer of f using a crossover operation of the fourth type described above and a mutation operation of the second type above. With a population size of 20, we apply 50 iterations of the genetic algorithm. As before, we used parameter values of $p_c = 0.75$ and $p_m = 0.0075$. Figure 14.12 shows plots of the best, average, and worst objective function values in the population for every iteration (generation) of the algorithm. The best-so-far solution obtained at the end of the 50 iterations is $[-0.0096, 1.5845]^\top$, with objective function value 8.1061, which is close to the result of Example 14.3. ■

EXERCISES

14.1 Write a MATLAB program to implement the Nelder-Mead algorithm applied to minimizing the function

$$f(x_1, x_2) = (x_2 - x_1)^4 + 12x_1x_2 - x_1 + x_2 - 3$$

on $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : x_1, x_2 \in [-1, 1]\}$. Locate the iteration points on the level sets of f . Connect the successive points with lines to show clearly the progression of the optimization process. Test your program with two starting points:

$$\mathbf{x}^{(0)} = \begin{bmatrix} 0.55 \\ 0.7 \end{bmatrix} \quad \text{and} \quad \mathbf{x}^{(0)} = \begin{bmatrix} -0.9 \\ -0.5 \end{bmatrix}.$$

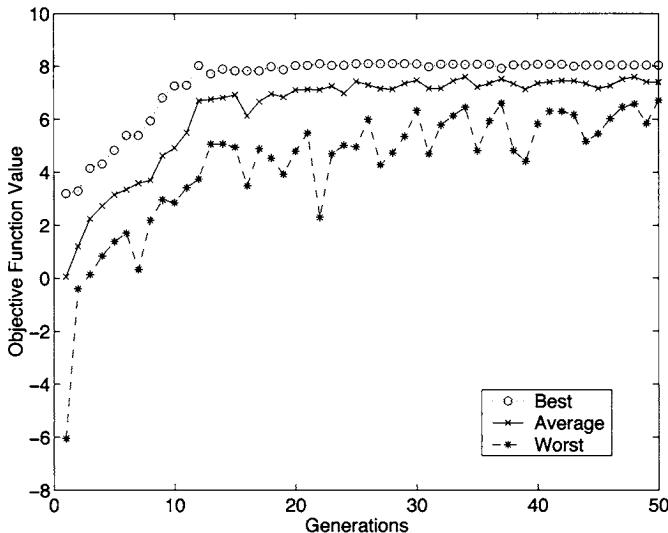


Figure 14.12 The best, average, and worst objective function values in the population for every iteration (generation) of the real-number genetic algorithm in Example 14.4.

14.2 Write MATLAB programs to implement naive random search and simulated annealing. Use the neighborhood

$$N(\mathbf{x}^{(k)}) = \{\mathbf{x} : x_i^{(k)} - \alpha \leq x_i \leq x_i^{(k)} + \alpha\},$$

where $\alpha > 0$ is prespecified, and pick $\mathbf{z}^{(k)}$ to be uniformly distributed on $N(\mathbf{x}^{(k)})$. Test both algorithms on maximizing the MATLAB “peaks” function given in Example 14.3. Observe the effect of varying α .

14.3 Write a MATLAB program to implement a particle swarm optimization algorithm. Test your implementation on maximizing the MATLAB “peaks” function given in Example 14.3.

14.4 This problem has four parts and is related to binary encoding for genetic algorithms.

- a. Let $(I)_{10}$ be the decimal representation for a given integer, and let $a_m a_{m-1} \cdots a_0$ be its binary representation; that is, each a_i is either 0 or 1, and

$$(I)_{10} = a_m 2^m + a_{m-1} 2^{m-1} + \cdots + a_1 2^1 + a_0 2^0.$$

Verify that the following is true:

$$(I)_{10} = (((\cdots (((a_m 2 + a_{m-1}) 2 + a_{m-2}) 2 + a_{m-3}) \cdots) 2 + a_1) 2 + a_0).$$

- b. The second expression in part a suggests a simple algorithm for converting from decimal representation to equivalent binary representation, as follows. Dividing both sides of the expression in part a by 2, the remainder is a_0 . Subsequent divisions by 2 yield the remaining bits a_1, a_2, \dots, a_m as remainders.

Use this algorithm to find the binary representation of the integer $(I)_{10} = 1995$.

- c. Let $(F)_{10}$ be the decimal representation for a given number in $[0, 1]$, and let $0.a_{-1}a_{-2}\dots$ be its binary representation, that is,

$$(F)_{10} = a_{-1}2^{-1} + a_{-2}2^{-2} + \dots$$

If this expression is multiplied by 2, the integer part of the product is a_{-1} . Subsequent multiplications yield the remaining bits a_{-2}, a_{-3}, \dots . As in part b, the above gives a simple algorithm for converting from a decimal fraction to its binary representations. Use this algorithm to find the binary representation of $(F)_{10} = 0.7265625$.

Note that we can combine the algorithms from parts b and c to convert an arbitrary positive decimal representation into its equivalent binary representation. Specifically, we apply the algorithms in parts b and c separately to the integer and fraction parts of the given decimal number, respectively.

- d. The procedure in part c may yield an infinitely long binary representation. If this is the case, then we need to determine the number of bits required to keep at least the same accuracy as the given decimal number. If we have a d -digit decimal fraction, then the number of bits b in the binary representation must satisfy $2^{-b} \leq 10^{-d}$, which yields $b \geq 3.32d$. Convert 19.95 to its equivalent binary representation with at least the same degree of accuracy (i.e., to two decimal places).

14.5 Given two chromosomes in a schema H , suppose that we swap some (or all) of the symbols between them at corresponding positions. Show that the resulting two chromosomes are also in H . From this fact we conclude that given two chromosomes in H , both offspring after the crossover operation are also in H . In other words, the crossover operation preserves membership in H .

14.6 Consider a two-point crossover scheme (see Example 14.2), described as follows. Given a pair of binary chromosomes of length L , we independently choose two random numbers, uniform over $1, \dots, L - 1$. We call the two numbers c_1 and c_2 , where $c_1 \leq c_2$. If $c_1 = c_2$, we do not swap any symbols (i.e., leave the two given parent chromosomes unchanged). If $c_1 < c_2$, we interchange the $(c_1 + 1)$ th through c_2 th bits in the given parent chromosomes.

Prove the analog of Lemma 14.2 for this case, given below.

Lemma: Given a chromosome in $M(k) \cap H$, the probability that it is chosen for crossover and neither of its offspring is in H is bounded above by

$$p_c \left(1 - \left(1 - \frac{l(H)}{L-1} \right)^2 \right).$$

□

Hint: Note that the two-point crossover operation is equivalent to a composition of two one-point crossover operations (i.e., doing two one-point crossover operations in succession).

14.7 State and prove the analog of Lemma 14.2 for an n -point crossover operation.

Hint: See Exercise 14.6.

14.8 Implement the roulette-wheel selection scheme using MATLAB.

Hint: Use the MATLAB functions `sum`, `cumsum`, and `find`.

14.9 Implement the crossover operation (one-point) using the MATLAB, assuming that we are given two binary parent chromosomes.

14.10 Implement the mutation operation using the MATLAB function `xor`, assuming that the chromosomes in the mating pool are binary vectors.

14.11 Write a MATLAB program to implement a genetic algorithm using binary encoding. Test your implementation on the following functions:

a. $f(x) = -15 \sin^2(2x) - (x-2)^2 + 160$, $|x| \leq 10$.

b. $f(x, y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10 \left(\frac{x}{5} - x^3 - y^5 \right) e^{-x^2-y^2} - \frac{e^{-(x+1)^2-y^2}}{3}$,
 $|x|, |y| \leq 3$ (considered in Example 14.3).

14.12 Write a MATLAB program to implement a real-number genetic algorithm. Test your implementation on the function $f(\mathbf{x}) = x_1 \sin(x_1) + x_2 \sin(5x_2)$ with the constraint set $\Omega = \{\mathbf{x} : 0 \leq x_1 \leq 10, 4 \leq x_2 \leq 6\}$.

