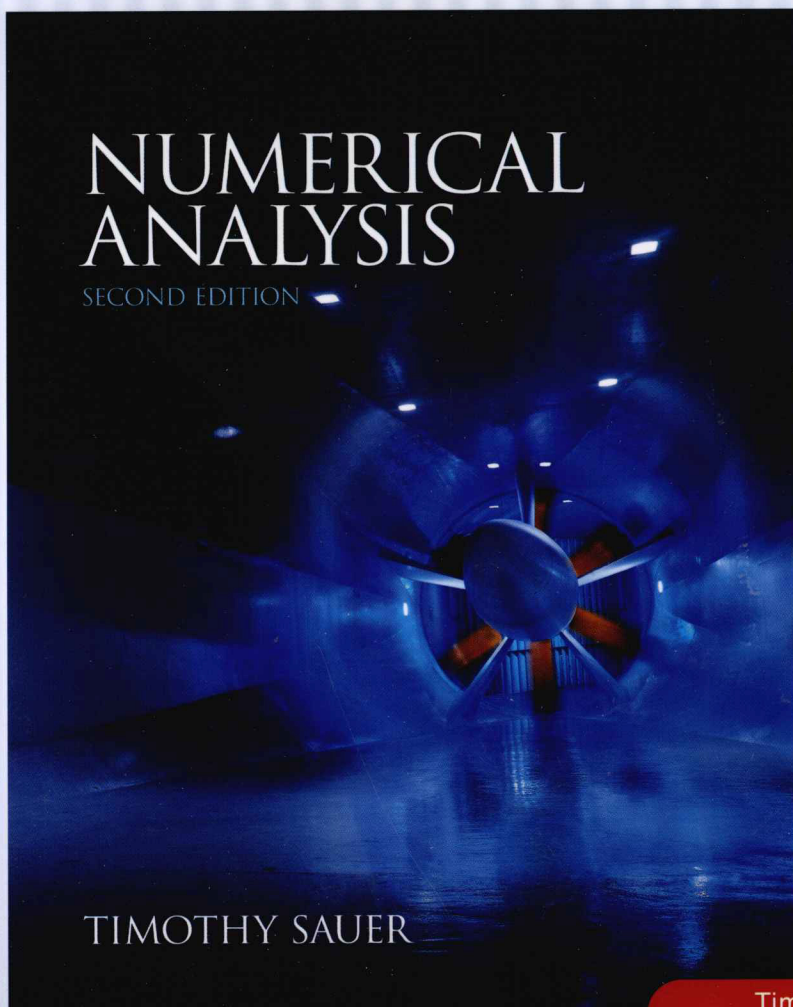


华章数学原版精品系列

数值分析

(英文版·第2版)



(美) Timothy Sauer 著
乔治梅森大学



机械工业出版社
China Machine Press

华章数字出版系列

数值分析

(英文版·第2版)

Numerical Analysis (Second Edition)

(美) Timothy Sauer 著
乔治梅森大学



机械工业出版社
China Machine Press

Original edition, entitled NUMERICAL ANALYSIS, 2E, 9780321783677 by SAUER, TIMOTHY, published by Pearson Education, Inc, publishing as Pearson, Copyright © 2012.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

English reprint edition published by PEARSON EDUCATION ASIA LTD., and CHINA MACHINE PRESS, Copyright © 2012.

This edition is manufactured in the People's Republic of China, and is authorized for sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macau SAR).

本书英文影印版由 Pearson Education Asia Ltd. 授权机械工业出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

仅限于中华人民共和国境内（不包括中国香港、澳门特别行政区和中国台湾地区）销售发行。

本书封面贴有 Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

封底无防伪标均为盗版

版权所有，侵权必究

本书法律顾问 北京市展达律师事务所

本书版权登记号：图字：01-2012-2645

图书在版编目（CIP）数据

数值分析（英文版·第2版）/（美）萨奥尔（Sauer, T.）著. —北京：机械工业出版社，2012.6
（华章数学原版精品系列）

书名原文：Numerical Analysis, Second Edition

ISBN 978-7-111-38582-0

I. 数… II. 萨… III. 数值分析—高等学校—教材—英文 IV. O241

中国版本图书馆 CIP 数据核字（2012）第 112002 号

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）

责任编辑：迟振春

北京诚信伟业印刷有限公司印刷

2012 年 6 月第 1 版第 1 次印刷

186mm×240mm·41.5 印张

标准书号：ISBN 978-7-111-38582-0

定价：89.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88378991；88361066

购书热线：(010) 68326294；88379649；68995259

投稿热线：(010) 88379604

读者信箱：hzsj@hzbook.com

Preface

Numerical Analysis is a text for students of engineering, science, mathematics, and computer science who have completed elementary calculus and matrix algebra. The primary goal is to construct and explore algorithms for solving science and engineering problems. The not-so-secret secondary mission is to help the reader locate these algorithms in a landscape of some potent and far-reaching principles. These unifying principles, taken together, constitute a dynamic field of current research and development in modern numerical and computational science.

The discipline of numerical analysis is jam-packed with useful ideas. Textbooks run the risk of presenting the subject as a bag of neat but unrelated tricks. For a deep understanding, readers need to learn much more than how to code Newton's Method, Runge-Kutta, and the Fast Fourier Transform. They must absorb the big principles, the ones that permeate numerical analysis and integrate its competing concerns of accuracy and efficiency.

The notions of convergence, complexity, conditioning, compression, and orthogonality are among the most important of the big ideas. Any approximation method worth its salt must converge to the correct answer as more computational resources are devoted to it, and the complexity of a method is a measure of its use of these resources. The conditioning of a problem, or susceptibility to error magnification, is fundamental to knowing how it can be attacked. Many of the newest applications of numerical analysis strive to realize data in a shorter or compressed way. Finally, orthogonality is crucial for efficiency in many algorithms, and is irreplaceable where conditioning is an issue or compression is a goal.

In this book, the roles of the five concepts in modern numerical analysis are emphasized in short thematic elements called Spotlights. They comment on the topic at hand and make informal connections to other expressions of the same concept elsewhere in the book. We hope that highlighting the five concepts in such an explicit way functions as a Greek chorus, accentuating what is really crucial about the theory on the page.

Although it is common knowledge that the ideas of numerical analysis are vital to the practice of modern science and engineering, it never hurts to be obvious. The Reality Checks provide concrete examples of the way numerical methods lead to solutions of important scientific and technological problems. These extended applications were chosen to be timely and close to everyday experience. Although it is impossible (and probably undesirable) to present the full details of the problems, the Reality Checks attempt to go deeply enough to show how a technique or algorithm can leverage a small amount of mathematics into a great payoff in technological design and function. The Reality Checks proved to be extremely popular as a source of student projects in the first edition, and have been extended and amplified in the second edition.

NEW TO THIS EDITION. The second edition features a major expansion of methods for solving systems of equations. The Cholesky factorization has been added to Chapter 2 for the solution of symmetric positive-definite matrix equations. For large linear systems, discussion of the Krylov approach, including the GMRES method, has been added to Chapter 4, along with new material on the use of preconditioners for symmetric and nonsymmetric problems. Modified Gram-Schmidt orthogonalization and the Levenberg-Marquardt Method are new to this edition. The treatment of PDEs in Chapter 8 has been extended to nonlinear PDEs, including reaction-diffusion equations and pattern formation. Expository material has been revised for greater readability based on feedback from students, and new exercises and computer problems have been added throughout.

TECHNOLOGY. The software package MATLAB is used both for exposition of algorithms and as a suggested platform for student assignments and projects. The amount of MATLAB code provided in the text is carefully modulated, due to the fact that too much

tends to be counterproductive. More MATLAB code is found in the early chapters, allowing the reader to gain proficiency in a gradual manner. Where more elaborate code is provided (in the study of interpolation, and ordinary and partial differential equations, for example), the expectation is for the reader to use what is given as a jumping-off point to exploit and extend.

It is not essential that any particular computational platform be used with this textbook, but the growing presence of MATLAB in engineering and science departments shows that a common language can smooth over many potholes. With MATLAB, all of the interface problems—data input/output, plotting, and so on—are solved in one fell swoop. Data structure issues (for example those that arise when studying sparse matrix methods) are standardized by relying on appropriate commands. MATLAB has facilities for audio and image file input and output. Differential equations simulations are simple to realize due to the animation commands built into MATLAB. These goals can all be achieved in other ways. But it is helpful to have one package that will run on almost all operating systems and simplify the details so that students can focus on the real mathematical issues. Appendix B is a MATLAB tutorial that can be used as a first introduction to students, or as a reference for those already familiar.

The text has a companion website, www.pearsonhighered.com/sauer, that contains the MATLAB programs taken directly from the text. In addition, new material and updates will be posted for users to download.

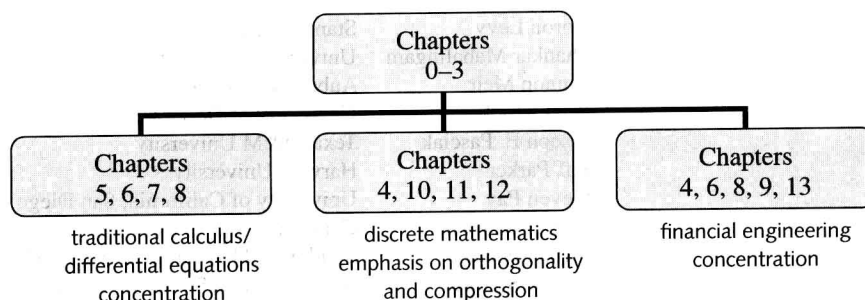
SUPPLEMENTS. To provide help for students, the **Student's Solutions Manual** (SSM: 0-321-78392) is available, with worked-out solutions to selected exercises. The **Instructor's Solutions Manual** (ISM: 0-321-783689) contains detailed solutions to the odd-numbered exercises, and answers to the even-numbered exercises. The manuals also show how to use MATLAB software as an aid to solving the types of problems that are presented in the Exercises and Computer Problems.

DESIGNING THE COURSE. *Numerical Analysis* is structured to move from foundational, elementary ideas at the outset to more sophisticated concepts later in the presentation. Chapter 0 provides fundamental building blocks for later use. Some instructors like to start at the beginning; others (including the author) prefer to start at Chapter 1 and fold in topics from Chapter 0 when required. Chapters 1 and 2 cover equation-solving in its various forms. Chapters 3 and 4 primarily treat the fitting of data, interpolation and least squares methods. In chapters 5–8, we return to the classical numerical analysis areas of continuous mathematics: numerical differentiation and integration, and the solution of ordinary and partial differential equations with initial and boundary conditions.

Chapter 9 develops random numbers in order to provide complementary methods to Chapters 5–8: the Monte-Carlo alternative to the standard numerical integration schemes and the counterpoint of stochastic differential equations are necessary when uncertainty is present in the model.

Compression is a core topic of numerical analysis, even though it often hides in plain sight in interpolation, least squares, and Fourier analysis. Modern compression techniques are featured in Chapters 10 and 11. In the former, the Fast Fourier Transform is treated as a device to carry out trigonometric interpolation, both in the exact and least squares sense. Links to audio compression are emphasized, and fully carried out in Chapter 11 on the Discrete Cosine Transform, the standard workhorse for modern audio and image compression. Chapter 12 on eigenvalues and singular values is also written to emphasize its connections to data compression, which are growing in importance in contemporary applications. Chapter 13 provides a short introduction to optimization techniques.

Numerical Analysis can also be used for a one-semester course with judicious choice of topics. Chapters 0–3 are fundamental for any course in the area. Separate one-semester tracks can be designed as follows:



ACKNOWLEDGMENTS

The second edition owes a debt to many people, including the students of many classes who have read and commented on earlier versions. In addition, Paul Lorcak, Maurino Bautista, and Tom Wegleitner were essential in helping me avoid embarrassing blunders. Suggestions from Nicholas Allgaier, Regan Beckham, Paul Calamai, Mark Friedman, David Hiebeler, Ashwani Kapila, Andrew Knyazev, Bo Li, Yijang Li, Jeff Parker, Robert Sachs, Evelyn Sander, Gantumur Tsogtgerel, and Thomas Wanner were greatly appreciated. The resourceful staff at Pearson, including William Hoffman, Caroline Celano, Beth Houston, Jeff Weidenaar, and Brandon Rawsley, as well as Shiny Rajesh at Integra-PDY, made the production of the second edition almost enjoyable. Finally, thanks are due to the helpful readers from other universities for their encouragement of this project and indispensable advice for improvement of earlier versions:

Eugene Allgower	Colorado State University
Constantin Bacuta	University of Delaware
Michele Benzi	Emory University
Jerry Bona	University of Illinois at Chicago
George Davis	Georgia State University
Chris Danforth	University of Vermont
Alberto Delgado	Bradley University
Robert Dillon	Washington State University
Qiang Du	Pennsylvania State University
Ahmet Duran	University of Michigan, Ann Arbor
Gregory Goeckel	Presbyterian College
Herman Gollwitzer	Drexel University
Don Hardcastle	Baylor University
David R. Hill	Temple University
Hideaki Kaneko	Old Dominion University
Daniel Kaplan	Macalester College
Fritz Keinert	Iowa State University
Akhtar A. Khan	Rochester Institute of Technology
Lucia M. Kimball	Bentley College
Colleen M. Kirk	California Polytechnic State University
Seppo Korpela	Ohio State University
William Layton	University of Pittsburgh
Brenton LeMesurier	College of Charleston
Melvin Leok	University of California, San Diego

Doron Levy	Stanford University
Shankar Mahalingam	University of California, Riverside
Amnon Meir	Auburn University
Peter Monk	University of Delaware
Joseph E. Pasciak	Texas A&M University
Jeff Parker	Harvard University
Steven Pav	University of California, San Diego
Jacek Polewczak	California State University
Jorge Rebaza	Southwest Missouri State University
Jeffrey Scroggs	North Carolina State University
Sergei Suslov	Arizona State University
Daniel Szyld	Temple University
Ahlam Tannouri	Morgan State University
Jin Wang	Old Dominion University
Bruno Welfert	Arizona State University
Nathaniel Whitaker	University of Massachusetts

Contents

PREFACE	iii
CHAPTER 0 Fundamentals	1
0.1 Evaluating a Polynomial	1
0.2 Binary Numbers	5
0.2.1 Decimal to binary	6
0.2.2 Binary to decimal	7
0.3 Floating Point Representation of Real Numbers	8
0.3.1 Floating point formats	8
0.3.2 Machine representation	11
0.3.3 Addition of floating point numbers	13
0.4 Loss of Significance	16
0.5 Review of Calculus	19
Software and Further Reading	23
CHAPTER 1 Solving Equations	24
1.1 The Bisection Method	25
1.1.1 Bracketing a root	25
1.1.2 How accurate and how fast?	28
1.2 Fixed-Point Iteration	30
1.2.1 Fixed points of a function	31
1.2.2 Geometry of Fixed-Point Iteration	33
1.2.3 Linear convergence of Fixed-Point Iteration	34
1.2.4 Stopping criteria	40
1.3 Limits of Accuracy	43
1.3.1 Forward and backward error	44
1.3.2 The Wilkinson polynomial	47
1.3.3 Sensitivity of root-finding	48
1.4 Newton's Method	51
1.4.1 Quadratic convergence of Newton's Method	53
1.4.2 Linear convergence of Newton's Method	55
1.5 Root-Finding without Derivatives	61
1.5.1 Secant Method and variants	61
1.5.2 Brent's Method	64
Reality Check 1: Kinematics of the Stewart platform	67
Software and Further Reading	69
CHAPTER 2 Systems of Equations	71
2.1 Gaussian Elimination	71
2.1.1 Naive Gaussian elimination	72
2.1.2 Operation counts	74

2.2	The LU Factorization	79
2.2.1	Matrix form of Gaussian elimination	79
2.2.2	Back substitution with the LU factorization	81
2.2.3	Complexity of the LU factorization	83
2.3	Sources of Error	85
2.3.1	Error magnification and condition number	86
2.3.2	Swamping	91
2.4	The PA = LU Factorization	95
2.4.1	Partial pivoting	95
2.4.2	Permutation matrices	97
2.4.3	PA = LU factorization	98
	Reality Check 2: The Euler–Bernoulli Beam	102
2.5	Iterative Methods	106
2.5.1	Jacobi Method	106
2.5.2	Gauss–Seidel Method and SOR	108
2.5.3	Convergence of iterative methods	111
2.5.4	Sparse matrix computations	113
2.6	Methods for symmetric positive-definite matrices	117
2.6.1	Symmetric positive-definite matrices	117
2.6.2	Cholesky factorization	119
2.6.3	Conjugate Gradient Method	121
2.6.4	Preconditioning	126
2.7	Nonlinear Systems of Equations	130
2.7.1	Multivariate Newton’s Method	131
2.7.2	Broyden’s Method	133
	Software and Further Reading	137

CHAPTER 3 Interpolation 138

3.1	Data and Interpolating Functions	139
3.1.1	Lagrange interpolation	140
3.1.2	Newton’s divided differences	141
3.1.3	How many degree d polynomials pass through n points?	144
3.1.4	Code for interpolation	145
3.1.5	Representing functions by approximating polynomials	147
3.2	Interpolation Error	151
3.2.1	Interpolation error formula	151
3.2.2	Proof of Newton form and error formula	153
3.2.3	Runge phenomenon	155
3.3	Chebyshev Interpolation	158
3.3.1	Chebyshev’s theorem	158
3.3.2	Chebyshev polynomials	160
3.3.3	Change of interval	162
3.4	Cubic Splines	166
3.4.1	Properties of splines	167
3.4.2	Endpoint conditions	173
3.5	Bézier Curves	179
	Reality Check 3: Fonts from Bézier curves	183
	Software and Further Reading	187

CHAPTER 4	Least Squares	188
4.1	Least Squares and the Normal Equations	188
4.1.1	Inconsistent systems of equations	189
4.1.2	Fitting models to data	193
4.1.3	Conditioning of least squares	197
4.2	A Survey of Models	201
4.2.1	Periodic data	201
4.2.2	Data linearization	203
4.3	QR Factorization	212
4.3.1	Gram–Schmidt orthogonalization and least squares	212
4.3.2	Modified Gram–Schmidt orthogonalization	218
4.3.3	Householder reflectors	220
4.4	Generalized Minimum Residual (GMRES) Method	225
4.4.1	Krylov methods	226
4.4.2	Preconditioned GMRES	228
4.5	Nonlinear Least Squares	230
4.5.1	Gauss–Newton Method	230
4.5.2	Models with nonlinear parameters	233
4.5.3	The Levenberg–Marquardt Method.	235
	Reality Check 4: GPS, Conditioning, and Nonlinear Least Squares	238
	Software and Further Reading	242
CHAPTER 5	Numerical Differentiation and Integration	243
5.1	Numerical Differentiation	244
5.1.1	Finite difference formulas	244
5.1.2	Rounding error	247
5.1.3	Extrapolation	249
5.1.4	Symbolic differentiation and integration	250
5.2	Newton–Cotes Formulas for Numerical Integration	254
5.2.1	Trapezoid Rule	255
5.2.2	Simpson’s Rule	257
5.2.3	Composite Newton–Cotes formulas	259
5.2.4	Open Newton–Cotes Methods	262
5.3	Romberg Integration	265
5.4	Adaptive Quadrature	269
5.5	Gaussian Quadrature	273
	Reality Check 5: Motion Control in Computer-Aided Modeling	278
	Software and Further Reading	280
CHAPTER 6	Ordinary Differential Equations	281
6.1	Initial Value Problems	282
6.1.1	Euler’s Method	283
6.1.2	Existence, uniqueness, and continuity for solutions	287
6.1.3	First-order linear equations	290
6.2	Analysis of IVP Solvers	293
6.2.1	Local and global truncation error	293

6.2.2	The explicit Trapezoid Method	297
6.2.3	Taylor Methods	300
6.3	Systems of Ordinary Differential Equations	303
6.3.1	Higher order equations	304
6.3.2	Computer simulation: the pendulum	305
6.3.3	Computer simulation: orbital mechanics	309
6.4	Runge–Kutta Methods and Applications	314
6.4.1	The Runge–Kutta family	314
6.4.2	Computer simulation: the Hodgkin–Huxley neuron	317
6.4.3	Computer simulation: the Lorenz equations	319
	Reality Check 6: The Tacoma Narrows Bridge	322
6.5	Variable Step-Size Methods	325
6.5.1	Embedded Runge–Kutta pairs	325
6.5.2	Order 4/5 methods	328
6.6	Implicit Methods and Stiff Equations	332
6.7	Multistep Methods	336
6.7.1	Generating multistep methods	336
6.7.2	Explicit multistep methods	339
6.7.3	Implicit multistep methods	342
	Software and Further Reading	347
CHAPTER 7	Boundary Value Problems	348
7.1	Shooting Method	349
7.1.1	Solutions of boundary value problems	349
7.1.2	Shooting Method implementation	352
	Reality Check 7: Buckling of a Circular Ring	355
7.2	Finite Difference Methods	357
7.2.1	Linear boundary value problems	357
7.2.2	Nonlinear boundary value problems	359
7.3	Collocation and the Finite Element Method	365
7.3.1	Collocation	365
7.3.2	Finite elements and the Galerkin Method	367
	Software and Further Reading	373
CHAPTER 8	Partial Differential Equations	374
8.1	Parabolic Equations	375
8.1.1	Forward Difference Method	375
8.1.2	Stability analysis of Forward Difference Method	379
8.1.3	Backward Difference Method	380
8.1.4	Crank–Nicolson Method	385
8.2	Hyperbolic Equations	393
8.2.1	The wave equation	393
8.2.2	The CFL condition	395
8.3	Elliptic Equations	398
8.3.1	Finite Difference Method for elliptic equations	399
	Reality Check 8: Heat distribution on a cooling fin	403
8.3.2	Finite Element Method for elliptic equations	406

8.4	Nonlinear partial differential equations	417
8.4.1	Implicit Newton solver	417
8.4.2	Nonlinear equations in two space dimensions	423
	Software and Further Reading	430
CHAPTER 9	Random Numbers and Applications	431
9.1	Random Numbers	432
9.1.1	Pseudo-random numbers	432
9.1.2	Exponential and normal random numbers	437
9.2	Monte Carlo Simulation	440
9.2.1	Power laws for Monte Carlo estimation	440
9.2.2	Quasi-random numbers	442
9.3	Discrete and Continuous Brownian Motion	446
9.3.1	Random walks	447
9.3.2	Continuous Brownian motion	449
9.4	Stochastic Differential Equations	452
9.4.1	Adding noise to differential equations	452
9.4.2	Numerical methods for SDEs	456
	Reality Check 9: The Black-Scholes Formula	464
	Software and Further Reading	465
CHAPTER 10	Trigonometric Interpolation and the FFT	467
10.1	The Fourier Transform	468
10.1.1	Complex arithmetic	468
10.1.2	Discrete Fourier Transform	470
10.1.3	The Fast Fourier Transform	473
10.2	Trigonometric Interpolation	476
10.2.1	The DFT Interpolation Theorem	476
10.2.2	Efficient evaluation of trigonometric functions	479
10.3	The FFT and Signal Processing	483
10.3.1	Orthogonality and interpolation	483
10.3.2	Least squares fitting with trigonometric functions	485
10.3.3	Sound, noise, and filtering	489
	Reality Check 10: The Wiener Filter	492
	Software and Further Reading	494
CHAPTER 11	Compression	495
11.1	The Discrete Cosine Transform	496
11.1.1	One-dimensional DCT	496
11.1.2	The DCT and least squares approximation	498
11.2	Two-Dimensional DCT and Image Compression	501
11.2.1	Two-dimensional DCT	501
11.2.2	Image compression	505
11.2.3	Quantization	508
11.3	Huffman Coding	514
11.3.1	Information theory and coding	514
11.3.2	Huffman coding for the JPEG format	517

11.4 Modified DCT and Audio Compression	519
11.4.1 Modified Discrete Cosine Transform	520
11.4.2 Bit quantization	525
Reality Check 11: A Simple Audio Codec	527
Software and Further Reading	530
 CHAPTER 12 Eigenvalues and Singular Values	531
12.1 Power Iteration Methods	531
12.1.1 Power Iteration	532
12.1.2 Convergence of Power Iteration	534
12.1.3 Inverse Power Iteration	535
12.1.4 Rayleigh Quotient Iteration	537
12.2 QR Algorithm	539
12.2.1 Simultaneous iteration	539
12.2.2 Real Schur form and the QR algorithm	542
12.2.3 Upper Hessenberg form	544
Reality Check 12: How Search Engines Rate Page Quality	549
12.3 Singular Value Decomposition	552
12.3.1 Finding the SVD in general	554
12.3.2 Special case: symmetric matrices	555
12.4 Applications of the SVD	557
12.4.1 Properties of the SVD	557
12.4.2 Dimension reduction	559
12.4.3 Compression	560
12.4.4 Calculating the SVD	561
Software and Further Reading	563
 CHAPTER 13 Optimization	565
13.1 Unconstrained Optimization without Derivatives	566
13.1.1 Golden Section Search	566
13.1.2 Successive parabolic interpolation	569
13.1.3 Nelder–Mead search	571
13.2 Unconstrained Optimization with Derivatives	575
13.2.1 Newton’s Method	576
13.2.2 Steepest Descent	577
13.2.3 Conjugate Gradient Search	578
Reality Check 13: Molecular Conformation and Numerical Optimization	580
Software and Further Reading	582
 Appendix A	583
A.1 Matrix Fundamentals	583
A.2 Block Multiplication	585
A.3 Eigenvalues and Eigenvectors	586
A.4 Symmetric Matrices	587
A.5 Vector Calculus	588

Appendix B	590
B.1 Starting MATLAB	590
B.2 Graphics	591
B.3 Programming in MATLAB	593
B.4 Flow Control	594
B.5 Functions	595
B.6 Matrix Operations	597
B.7 Animation and Movies	597
ANSWERS TO SELECTED EXERCISES	599
BIBLIOGRAPHY	626
INDEX	637



Fundamentals

This introductory chapter provides basic building blocks necessary for the construction and understanding of the algorithms of the book. They include fundamental ideas of introductory calculus and function evaluation, the details of machine arithmetic as it is carried out on modern computers, and discussion of the loss of significant digits resulting from poorly-designed calculations.

After discussing efficient methods for evaluating polynomials, we study the binary number system, the representation of floating point numbers and the common protocols used for rounding. The effects of the small rounding errors on computations are magnified in ill-conditioned problems. The battle to limit these pernicious effects is a recurring theme throughout the rest of the chapters.

The goal of this book is to present and discuss methods of solving mathematical problems with computers. The most fundamental operations of arithmetic are addition and multiplication. These are also the operations needed to evaluate a polynomial $P(x)$ at a particular value x . It is no coincidence that polynomials are the basic building blocks for many computational techniques we will construct.

Because of this, it is important to know how to evaluate a polynomial. The reader probably already knows how and may consider spending time on such an easy problem slightly ridiculous! But the more basic an operation is, the more we stand to gain by doing it right. Therefore we will think about how to implement polynomial evaluation as efficiently as possible.

0.1 EVALUATING A POLYNOMIAL

What is the best way to evaluate

$$P(x) = 2x^4 + 3x^3 - 3x^2 + 5x - 1,$$

say, at $x = 1/2$? Assume that the coefficients of the polynomial and the number $1/2$ are stored in memory, and try to minimize the number of additions and multiplications required

to get $P(1/2)$. To simplify matters, we will not count time spent storing and fetching numbers to and from memory.

METHOD 1 The first and most straightforward approach is

$$P\left(\frac{1}{2}\right) = 2 * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} + 3 * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} - 3 * \frac{1}{2} * \frac{1}{2} + 5 * \frac{1}{2} - 1 = \frac{5}{4}. \quad (0.1)$$

The number of multiplications required is 10, together with 4 additions. Two of the additions are actually subtractions, but because subtraction can be viewed as adding a negative stored number, we will not worry about the difference.

There surely is a better way than (0.1). Effort is being duplicated—operations can be saved by eliminating the repeated multiplication by the input $1/2$. A better strategy is to first compute $(1/2)^4$, storing partial products as we go. That leads to the following method:

METHOD 2 Find the powers of the input number $x = 1/2$ first, and store them for future use:

$$\begin{aligned} \frac{1}{2} * \frac{1}{2} &= \left(\frac{1}{2}\right)^2 \\ \left(\frac{1}{2}\right)^2 * \frac{1}{2} &= \left(\frac{1}{2}\right)^3 \\ \left(\frac{1}{2}\right)^3 * \frac{1}{2} &= \left(\frac{1}{2}\right)^4. \end{aligned}$$

Now we can add up the terms:

$$P\left(\frac{1}{2}\right) = 2 * \left(\frac{1}{2}\right)^4 + 3 * \left(\frac{1}{2}\right)^3 - 3 * \left(\frac{1}{2}\right)^2 + 5 * \frac{1}{2} - 1 = \frac{5}{4}.$$

There are now 3 multiplications of $1/2$, along with 4 other multiplications. Counting up, we have reduced to 7 multiplications, with the same 4 additions. Is the reduction from 14 to 11 operations a significant improvement? If there is only one evaluation to be done, then probably not. Whether Method 1 or Method 2 is used, the answer will be available before you can lift your fingers from the computer keyboard. However, suppose the polynomial needs to be evaluated at different inputs x several times per second. Then the difference may be crucial to getting the information when it is needed.

Is this the best we can do for a degree 4 polynomial? It may be hard to imagine that we can eliminate three more operations, but we can. The best elementary method is the following one:

METHOD 3 (Nested Multiplication) Rewrite the polynomial so that it can be evaluated from the inside out:

$$\begin{aligned} P(x) &= -1 + x(5 - 3x + 3x^2 + 2x^3) \\ &= -1 + x(5 + x(-3 + 3x + 2x^2)) \\ &= -1 + x(5 + x(-3 + x(3 + 2x))) \\ &= -1 + x * (5 + x * (-3 + x * (3 + x * 2))). \end{aligned} \quad (0.2)$$

Here the polynomial is written backwards, and powers of x are factored out of the rest of the polynomial. Once you can see to write it this way—no computation is required to do the rewriting—the coefficients are unchanged. Now evaluate from the inside out:

$$\begin{aligned}
&\text{multiply } \frac{1}{2} * 2, \quad \text{add } + 3 \rightarrow 4 \\
&\text{multiply } \frac{1}{2} * 4, \quad \text{add } - 3 \rightarrow -1 \\
&\text{multiply } \frac{1}{2} * -1, \quad \text{add } + 5 \rightarrow \frac{9}{2} \\
&\text{multiply } \frac{1}{2} * \frac{9}{2}, \quad \text{add } - 1 \rightarrow \frac{5}{4}.
\end{aligned} \tag{0.3}$$

This method, called **nested multiplication** or **Horner's method**, evaluates the polynomial in 4 multiplications and 4 additions. A general degree d polynomial can be evaluated in d multiplications and d additions. Nested multiplication is closely related to synthetic division of polynomial arithmetic.

The example of polynomial evaluation is characteristic of the entire topic of computational methods for scientific computing. First, computers are very fast at doing very simple things. Second, it is important to do even simple tasks as efficiently as possible, since they may be executed many times. Third, the best way may not be the obvious way. Over the last half-century, the fields of numerical analysis and scientific computing, hand in hand with computer hardware technology, have developed efficient solution techniques to attack common problems.

While the standard form for a polynomial $c_1 + c_2x + c_3x^2 + c_4x^3 + c_5x^4$ can be written in nested form as

$$c_1 + x(c_2 + x(c_3 + x(c_4 + x(c_5))))), \tag{0.4}$$

some applications require a more general form. In particular, interpolation calculations in Chapter 3 will require the form

$$c_1 + (x - r_1)(c_2 + (x - r_2)(c_3 + (x - r_3)(c_4 + (x - r_4)(c_5)))), \tag{0.5}$$

where we call r_1, r_2, r_3 , and r_4 the **base points**. Note that setting $r_1 = r_2 = r_3 = r_4 = 0$ in (0.5) recovers the original nested form (0.4).

The following MATLAB code implements the general form of nested multiplication (compare with (0.3)):

```

%Program 0.1 Nested multiplication
%Evaluates polynomial from nested form using Horner's Method
%Input: degree d of polynomial,
%       array of d+1 coefficients c (constant term first),
%       x-coordinate x at which to evaluate, and
%       array of d base points b, if needed
%Output: value y of polynomial at x
function y=nest(d,c,x,b)
if nargin<4, b=zeros(d,1); end
y=c(d+1);
for i=d:-1:1
    y = y.*(x-b(i))+c(i);
end

```

Running this MATLAB function is a matter of substituting the input data, which consist of the degree, coefficients, evaluation points, and base points. For example, polynomial (0.2) can be evaluated at $x = 1/2$ by the MATLAB command