Texts in Computational Science and Engineering 11

Editors

For further volumes:
http://www.springer.com/series/5151

Walter Gander • Martin J. Gander • Felix Kwok

Scientific Computing

An Introduction using Maple and MATLAB

Springer

Walter Gander
Departement Informatik
ETH Zürich
Zürich
Switzerland

Martin J. Gander
Felix Kwok
Section de Mathématiques
Université de Genève
Genève
Switzerland

ISSN 1611-0994
ISBN 978-3-319-04324-1 ISBN 978-3-319-04325-8 (eBook)
DOI 10.1007/978-3-319-04325-8
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014937000

Mathematics Subject Classification (2010): 65-00, 65-01

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

This book is dedicated to
Professor Gene H. Golub
1932–2007

(picture by Jill Knuth)

The three authors represent three generations of mathematicians who have been enormously influenced by Gene Golub.
He shaped our lives and our academic careers through his advice, his leadership, his friendship and his care for younger scientists.
We are indebted and will always honor his memory.

Preface

We are conducting ever more complex computations built upon the assumption that the underlying numerical methods are mature and reliable.
When we bundle existing algorithms into libraries and wrap them into packages to facilitate easy use, we create de facto standards that make it easy to ignore numerical analysis.

John Guckenheimer, president SIAM, in *SIAM News, June 1998: Numerical Computation in the Information Age*

When redrafting the book I was tempted to present the algorithms in ALGOL, but decided that the difficulties of providing procedures which were correct in every detail were prohibitive at this stage.

James Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, 1988.

This book is an introduction to *scientific computing*, the mathematical modeling in science and engineering and the study of how to exploit computers in the solution of technical and scientific problems. It is based on mathematics, numerical and symbolic/algebraic computations, parallel/distributed processing and visualization. It is also a popular and growing area — many new curricula in *computational science and engineering* have been, and continue to be, developed, leading to new academic degrees and even entire new disciplines.

A prerequisite for this development is the ubiquitous presence of computers, which are being used by virtually every student and scientist. While traditional scientific work is based on developing theories and performing experiments, the possibility to use computers at any time has created a third way of increasing our knowledge, which is through modeling and simulation. The use of simulation is further facilitated by the availability of sophisticated, robust and easy-to-use software libraries. This has the obvious advantage of shielding the user from the underlying numerics; however, this also has the danger of leaving the user unaware of the limitations of the algorithms, which can lead to incorrect results when used improperly. Moreover, some algorithms can be fast for certain types of problems but highly inefficient for others. Thus, it is important for the user to be able to make an informed decision on which algorithms to use, based on the properties of the problem to be solved. The goal of this book is to familiarize the reader with the basic

concepts of scientific computing and algorithms that form the workhorses of many numerical libraries. In fact, we will also emphasize the effective implementation of the algorithms discussed.

Numerical scientific computing has a long history; in fact, computers were first built for this purpose. *Konrad Zuse* [154] built his first (mechanical) computer in 1938 because he wanted to have a machine that would solve systems of linear equations that arise, e.g., when a civil engineer designs a bridge. At about the same time (and independently), *Howard H. Aiken* wanted to build a machine that would solve systems of ordinary differential equations [17].

The first high quality software libraries contained indeed numerical algorithms. They were produced in an international effort in the programming language ALGOL 60 [111], and are described in the handbook "Numerical Algebra" [148]. These fundamental procedures for solving linear equations and eigenvalue problems were developed further, rewritten in FORTRAN, and became the LINPACK [26] and EISPACK [47] libraries. They are still in use and available at `www.netlib.org` from Netlib. In order to help students to use this software, Cleve Moler created around 1980 a friendly interface to those subroutines, which he called MATLAB (Matrix Laboratory). MATLAB was so successful that a company was founded: MathWorks. Today, MATLAB is "the language of technical computing", a very powerful tool in scientific computing.

Parallel to the development of numerical libraries, there were also efforts to do exact and algebraic computations. The first computer algebra systems were created some 50 years ago: At ETH, Max Engeli created SYMBAL, and at MIT, Joel Moses MACSYMA. MACSYMA is the oldest system that is still available. However, computer algebra computations require much more computer resources than numerical calculations. Therefore, only when computers became more powerful did these systems flourish. Today the market leaders are MATHEMATICA and MAPLE.

Often, a problem may be solved analytically ("exactly") by a computer algebra system. In general, however, analytical solutions do not exist, and numerical approximations or other special techniques must be used instead. Moreover, computer Algebra is a very powerful tool for deriving numerical algorithms; we use MAPLE for this purpose in several chapters of this book. Thus, computer algebra systems and numerical libraries are complementary tools: working with both is essential in scientific computing. We have chosen MATLAB and MAPLE as basic tools for this book. Nonetheless, we are aware that the difference between pure computer algebra systems and numerical MATLAB-like systems is disappearing, and the two may merge and become indistinguishable by the user in the near future.

How to use this book

Prerequisites for understanding this book are courses in calculus and linear algebra. The content of this book is too much for a typical one semester course in scientific computing. However, the instructor can choose those sections that he wishes to teach and that fit his schedule. For example, for an introductory course in scientific computing, one can very well use the least squares chapter and teach only one of the methods for computing the QR decomposition. However, for an advanced course focused solely on least squares methods, one may also wish to consider the singular value decomposition (SVD) as a computational tool for solving least squares problems. In this case, the book also provides a detailed description on how to compute the SVD in the chapter on eigenvalues. The material is presented in such a way that a student can also learn directly from the book. To help the reader navigate the volume, we provide in section 1.2 some sample courses that have been taught by the authors at various institutions.

The focus of the book is algorithms: we would like to explain to the students how some fundamental functions in mathematical software are designed. Many exercises require programming in MATLAB or MAPLE, since we feel it is important for students to gain experience in using such powerful software systems. They should also know about their limitations and be aware of the issue addressed by John Guckenheimer. We tried to include meaningful examples and problems, not just academic exercises.

Acknowledgments

The authors would like to thank Oscar Chinellato, Ellis Whitehead, Oliver Ernst and Laurence Halpern for their careful proofreading and helpful suggestions.

Walter Gander is indebted to Hong Kong Baptist University (HKBU) and especially to its Vice President Academic, Franklin Luk, for giving him the opportunity to continue to teach students after his retirement at ETH. Several chapters of this book have been presented and improved successfully in courses at HKBU. We are also thankful to the University of Geneva, where we met many times to finalize the manuscript.

Geneva and Zürich, August 2013

Walter Gander, Martin J. Gander, Felix Kwok

Contents

Chapter 1. Why Study Scientific Computing? 1
1.1 Example: Designing a Suspension Bridge 1
1.1.1 Constructing a Model 1
1.1.2 Simulating the Bridge 3
1.1.3 Calculating Resonance Frequencies 4
1.1.4 Matching Simulations with Experiments 5
1.2 Navigating this Book: Sample Courses 6
1.2.1 A First Course in Numerical Analysis 7
1.2.2 Advanced Courses 8
1.2.3 Dependencies Between Chapters 8

Chapter 2. Finite Precision Arithmetic 9
2.1 Introductory Example 10
2.2 Real Numbers and Machine Numbers 11
2.3 The IEEE Standard 14
2.3.1 Single Precision 14
2.3.2 Double Precision 16
2.4 Rounding Errors 19
2.4.1 Standard Model of Arithmetic 19
2.4.2 Cancellation 20
2.5 Condition of a Problem 24
2.5.1 Norms 24
2.5.2 Big- and Little-O Notation 27
2.5.3 Condition Number 29
2.6 Stable and Unstable Algorithms 33
2.6.1 Forward Stability 33
2.6.2 Backward Stability 36
2.7 Calculating with Machine Numbers: Tips and Tricks 38
2.7.1 Associative Law 38
2.7.2 Summation Algorithm by W. Kahan 39
2.7.3 Small Numbers 40
2.7.4 Monotonicity 40
2.7.5 Avoiding Overflow 41

2.7.6 Testing for Overflow . . . 42
2.7.7 Avoiding Cancellation . . . 43
2.7.8 Computation of Mean and Standard Deviation . . . 45
2.8 Stopping Criteria . . . 48
2.8.1 Machine-independent Algorithms . . . 48
2.8.2 Test Successive Approximations . . . 51
2.8.3 Check the Residual . . . 51
2.9 Problems . . . 52

Chapter 3. Linear Systems of Equations . . . 61
3.1 Introductory Example . . . 62
3.2 Gaussian Elimination . . . 66
3.2.1 LU Factorization . . . 73
3.2.2 Backward Stability . . . 77
3.2.3 Pivoting and Scaling . . . 79
3.2.4 Sum of Rank-One Matrices . . . 82
3.3 Condition of a System of Linear Equations . . . 84
3.4 Cholesky Decomposition . . . 88
3.4.1 Symmetric Positive Definite Matrices . . . 88
3.4.2 Stability and Pivoting . . . 92
3.5 Elimination with Givens Rotations . . . 95
3.6 Banded matrices . . . 97
3.6.1 Storing Banded Matrices . . . 97
3.6.2 Tridiagonal Systems . . . 99
3.6.3 Solving Banded Systems with Pivoting . . . 100
3.6.4 Using Givens Rotations . . . 103
3.7 Problems . . . 105

Chapter 4. Interpolation . . . 113
4.1 Introductory Examples . . . 114
4.2 Polynomial Interpolation . . . 116
4.2.1 Lagrange Polynomials . . . 117
4.2.2 Interpolation Error . . . 119
4.2.3 Barycentric Formula . . . 121
4.2.4 Newton's Interpolation Formula . . . 123
4.2.5 Interpolation Using Orthogonal Polynomials . . . 127
4.2.6 Change of Basis, Relation with LU and QR . . . 132
4.2.7 Aitken-Neville Interpolation . . . 139
4.2.8 Extrapolation . . . 142
4.3 Piecewise Interpolation with Polynomials . . . 144
4.3.1 Classical Cubic Splines . . . 145
4.3.2 Derivatives for the Spline Function . . . 147
4.3.3 Sherman–Morrison–Woodbury Formula . . . 155
4.3.4 Spline Curves . . . 157

4.4 Trigonometric Interpolation 158
4.4.1 Trigonometric Polynomials 160
4.4.2 Fast Fourier Transform (FFT) 162
4.4.3 Trigonometric Interpolation Error 164
4.4.4 Convolutions Using FFT 168
4.5 Problems 171

Chapter 5. Nonlinear Equations 181
5.1 Introductory Example 182
5.2 Scalar Nonlinear Equations 184
5.2.1 Bisection 185
5.2.2 Fixed Point Iteration 187
5.2.3 Convergence Rates 190
5.2.4 Aitken Acceleration and the ε-Algorithm 193
5.2.5 Construction of One Step Iteration Methods 199
5.2.6 Multiple Zeros 205
5.2.7 Multi-Step Iteration Methods 207
5.2.8 A New Iteration Formula 210
5.2.9 Dynamical Systems 212
5.3 Zeros of Polynomials 215
5.3.1 Condition of the Zeros 217
5.3.2 Companion Matrix 220
5.3.3 Horner's Scheme 222
5.3.4 Number Conversions 227
5.3.5 Newton's Method: Classical Version 230
5.3.6 Newton Method Using Taylor Expansions 231
5.3.7 Newton Method for Real Simple Zeros 232
5.3.8 Nickel's Method 237
5.3.9 Laguerre's Method 239
5.4 Nonlinear Systems of Equations 240
5.4.1 Fixed Point Iteration 242
5.4.2 Theorem of Banach 242
5.4.3 Newton's Method 245
5.4.4 Continuation Methods 251
5.5 Problems 252

Chapter 6. Least Squares Problems 261
6.1 Introductory Examples 262
6.2 Linear Least Squares Problem and the Normal Equations 266
6.3 Singular Value Decomposition (SVD) 269
6.3.1 Pseudoinverse 274
6.3.2 Fundamental Subspaces 275
6.3.3 Solution of the Linear Least Squares Problem 277
6.3.4 SVD and Rank 279

6.4 Condition of the Linear Least Squares Problem 280
6.4.1 Differentiation of Pseudoinverses 282
6.4.2 Sensitivity of the Linear Least Squares Problem 285
6.4.3 Normal Equations and Condition 286
6.5 Algorithms Using Orthogonal Matrices 287
6.5.1 QR Decomposition 287
6.5.2 Method of Householder 289
6.5.3 Method of Givens 292
6.5.4 Fast Givens 298
6.5.5 Gram-Schmidt Orthogonalization 301
6.5.6 Gram-Schmidt with Reorthogonalization 306
6.5.7 Partial Reorthogonalization 308
6.5.8 Updating and Downdating the QR Decomposition 311
6.5.9 Covariance Matrix Computations Using QR 320
6.6 Linear Least Squares Problems with Linear Constraints . . . 323
6.6.1 Solution with SVD 325
6.6.2 Classical Solution Using Lagrange Multipliers 328
6.6.3 Direct Elimination of the Constraints 330
6.6.4 Null Space Method 333
6.7 Special Linear Least Squares Problems with Quadratic Constraint 334
6.7.1 Fitting Lines 335
6.7.2 Fitting Ellipses 337
6.7.3 Fitting Hyperplanes, Collinearity Test 340
6.7.4 Procrustes or Registration Problem 344
6.7.5 Total Least Squares 349
6.8 Nonlinear Least Squares Problems 354
6.8.1 Notations and Definitions 354
6.8.2 Newton's Method 356
6.8.3 Gauss-Newton Method 360
6.8.4 Levenberg-Marquardt Algorithm 361
6.9 Least Squares Fit with Piecewise Functions 364
6.9.1 Structure of the Linearized Problem 367
6.9.2 Piecewise Polynomials 368
6.9.3 Examples 372
6.10 Problems 374

Chapter 7. Eigenvalue Problems 387
7.1 Introductory Example 388
7.2 A Brief Review of the Theory 392
7.2.1 Eigen-Decomposition of a Matrix 392
7.2.2 Characteristic Polynomial 396
7.2.3 Similarity Transformations 396

7.2.4 Diagonalizable Matrices . . . 397
7.2.5 Exponential of a Matrix . . . 397
7.2.6 Condition of Eigenvalues . . . 398
7.3 Method of Jacobi . . . 405
7.3.1 Reducing Cost by Using Symmetry . . . 414
7.3.2 Stopping Criterion . . . 417
7.3.3 Algorithm of Rutishauser . . . 417
7.3.4 Remarks and Comments on `Jacobi` . . . 420
7.4 Power Methods . . . 422
7.4.1 Power Method . . . 423
7.4.2 Inverse Power Method (Shift-and-Invert) . . . 424
7.4.3 Orthogonal Iteration . . . 425
7.5 Reduction to Simpler Form . . . 429
7.5.1 Computing Givens Rotations . . . 429
7.5.2 Reduction to Hessenberg Form . . . 430
7.5.3 Reduction to Tridiagonal Form . . . 434
7.6 QR Algorithm . . . 436
7.6.1 Some History . . . 437
7.6.2 QR Iteration . . . 437
7.6.3 Basic Facts . . . 437
7.6.4 Preservation of Form . . . 438
7.6.5 Symmetric Tridiagonal Matrices . . . 439
7.6.6 Implicit QR Algorithm . . . 443
7.6.7 Convergence of the QR Algorithm . . . 445
7.6.8 Wilkinson's Shift . . . 447
7.6.9 Test for Convergence and Deflation . . . 448
7.6.10 Unreduced Matrices have Simple Eigenvalues . . . 449
7.6.11 Specific Numerical Examples . . . 451
7.6.12 Computing the Eigenvectors . . . 453
7.7 Computing the Singular Value Decomposition (SVD) . . . 453
7.7.1 Transformations . . . 454
7.7.2 Householder-Rutishauser Bidiagonalization . . . 454
7.7.3 Golub-Kahan-Lanczos Bidiagonalization . . . 457
7.7.4 Eigenvalues and Singular Values . . . 457
7.7.5 Algorithm of Golub-Reinsch . . . 458
7.8 QD Algorithm . . . 464
7.8.1 Progressive QD Algorithm . . . 464
7.8.2 Orthogonal LR-Cholesky Algorithm . . . 468
7.8.3 Differential QD Algorithm . . . 472
7.8.4 Improving Convergence Using Shifts . . . 474
7.8.5 Connection to Orthogonal Decompositions . . . 478
7.9 Problems . . . 482

Chapter 8. Differentiation 487
8.1 Introductory Example 488
8.2 Finite Differences 491
8.2.1 Generating Finite Difference Approximations 494
8.2.2 Discrete Operators for Partial Derivatives 496
8.3 Algorithmic Differentiation 499
8.3.1 Idea Behind Algorithmic Differentiation 499
8.3.2 Rules for Algorithmic Differentiation 504
8.3.3 Example: Circular Billiard 505
8.3.4 Example: Nonlinear Eigenvalue Problems 509
8.4 Problems 514

Chapter 9. Quadrature 517
9.1 Computer Algebra and Numerical Approximations 518
9.2 Newton–Cotes Rules 521
9.2.1 Error of Newton–Cotes Rules 525
9.2.2 Composite Rules 527
9.2.3 Euler–Maclaurin Summation Formula 531
9.2.4 Romberg Integration 537
9.3 Gauss Quadrature 541
9.3.1 Characterization of Nodes and Weights 545
9.3.2 Orthogonal Polynomials 547
9.3.3 Computing the Weights 552
9.3.4 Golub–Welsch Algorithm 555
9.4 Adaptive Quadrature 561
9.4.1 Stopping Criterion 563
9.4.2 Adaptive Simpson quadrature 565
9.4.3 Adaptive Lobatto quadrature 569
9.5 Problems 577

Chapter 10. Numerical Ordinary Differential Equations . . . 583
10.1 Introductory Examples 584
10.2 Basic Notation and Solution Techniques 587
10.2.1 Notation, Existence of Solutions 587
10.2.2 Analytical and Numerical Solutions 589
10.2.3 Solution by Taylor Expansions 591
10.2.4 Computing with Power Series 593
10.2.5 Euler's Method 597
10.2.6 Autonomous ODE, Reduction to First Order System 603
10.3 Runge-Kutta Methods 604
10.3.1 Explicit Runge-Kutta Methods 604
10.3.2 Local Truncation Error 606
10.3.3 Order Conditions 608
10.3.4 Convergence 615

10.3.5 Adaptive Integration 617
10.3.6 Implicit Runge-Kutta Methods 625
10.4 Linear Multistep Methods 631
10.4.1 Local Truncation Error 635
10.4.2 Order Conditions 636
10.4.3 Zero Stability 638
10.4.4 Convergence 643
10.5 Stiff Problems 646
10.5.1 A-Stability 650
10.5.2 A Nonlinear Example 653
10.5.3 Differential Algebraic Equations 655
10.6 Geometric Integration 656
10.6.1 Symplectic Methods 658
10.6.2 Energy Preserving Methods 661
10.7 Delay Differential Equations 664
10.8 Problems 666

Chapter 11. Iterative Methods for Linear Systems 673
11.1 Introductory Example 675
11.2 Solution by Iteration 677
11.2.1 Matrix Splittings 677
11.2.2 Residual, Error and the Difference of Iterates 678
11.2.3 Convergence Criteria 680
11.2.4 Singular Systems 683
11.2.5 Convergence Factor and Convergence Rate 684
11.3 Classical Stationary Iterative Methods 687
11.3.1 Regular Splittings and M-Matrices 687
11.3.2 Jacobi 691
11.3.3 Gauss-Seidel 694
11.3.4 Successive Over-relaxation (SOR) 695
11.3.5 Richardson 702
11.4 Local Minimization by Nonstationary Iterative Methods . . . 704
11.4.1 Conjugate Residuals 705
11.4.2 Steepest Descent 705
11.5 Global Minimization with Chebyshev Polynomials 708
11.5.1 Chebyshev Semi-Iterative Method 719
11.5.2 Acceleration of SSOR 724
11.6 Global Minimization by Extrapolation 726
11.6.1 Minimal Polynomial Extrapolation (MPE) 729
11.6.2 Reduced Rank Extrapolation (RRE) 733
11.6.3 Modified Minimal Polynomial Extrapolation (MMPE) 734
11.6.4 Topological ε-Algorithm (TEA) 735
11.6.5 Recursive Topological ε-Algorithm 737

11.7 Krylov Subspace Methods 739
11.7.1 The Conjugate Gradient Method 740
11.7.2 Arnoldi Process 758
11.7.3 The Symmetric Lanczos Algorithm 761
11.7.4 Solving Linear Equations with Arnoldi 766
11.7.5 Solving Linear Equations with Lanczos 769
11.7.6 Generalized Minimum Residual: GMRES 773
11.7.7 Classical Lanczos for Non-Symmetric Matrices 780
11.7.8 Biconjugate Gradient Method (BiCG) 793
11.7.9 Further Krylov Methods 800
11.8 Preconditioning 801
11.9 Problems 804

Chapter 12. Optimization 817
12.1 Introductory Examples 818
12.1.1 How much daily exercise is optimal ? 818
12.1.2 Mobile Phone Networks 821
12.1.3 A Problem from Operations Research 828
12.1.4 Classification of Optimization Problems 831
12.2 Mathematical Optimization 832
12.2.1 Local Minima 832
12.2.2 Constrained minima and Lagrange multipliers 835
12.2.3 Equality and Inequality Constraints 838
12.3 Unconstrained Optimization 842
12.3.1 Line Search Methods 842
12.3.2 Trust Region Methods 856
12.3.3 Direct Methods 859
12.4 Constrained Optimization 862
12.4.1 Linear Programming 862
12.4.2 Penalty and Barrier Functions 872
12.4.3 Interior Point Methods 873
12.4.4 Sequential Quadratic Programming 877
12.5 Problems 880

Bibliography 887

Index 895