

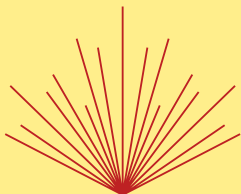
CMS/CAIMS Books in Mathematics



Canadian
Mathematical Society
Société mathématique
du Canada

David E. Stewart

Numerical Analysis: A Graduate Course



CAIMS
SCMAI

 Springer

CMS/CAIMS Books in Mathematics

Volume 4

Series Editors

Karl Dilcher, Department of Mathematics and Statistics, Dalhousie University, Halifax, NS, Canada

Frithjof Lutscher, Department of Mathematics, University of Ottawa, Ottawa, ON, Canada

Nilima Nigam, Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada

Keith Taylor, Department of Math and Statistics, Dalhousie University, Halifax, NS, Canada

Associate Editors

Ben Adcock, Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada

Martin Barlow, University of British Columbia, Vancouver, BC, Canada

Heinz H. Bauschke, University of British Columbia, Kelowna, BC, Canada

Matt Davison, Department of Statistics and Actuarial Science, Western University, London, ON, Canada

Leah Keshet, Department of Mathematics, University of British Columbia, Vancouver, BC, Canada

Niky Kamran, Mathematics & Statistics, McGill University, Montreal, QC, Canada

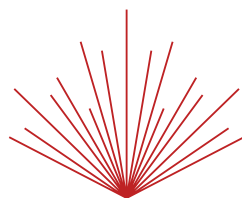
Mikhail Kotchetov, Quarter, Andrew Wiles Bldg, Memorial University of Newfoundland, St. John's, Canada

Raymond J. Spiteri, Department of Computer Science, University of Saskatchewan, Saskatoon, SK, Canada

CMS/CAIMS Books in Mathematics is a collection of monographs and graduate-level textbooks published in cooperation jointly with the Canadian Mathematical Society- Société mathématique du Canada and the Canadian Applied and Industrial Mathematics Society-Société Canadienne de Mathématiques Appliquées et Industrielles. This series offers authors the joint advantage of publishing with two major mathematical societies and with a leading academic publishing company. The series is edited by Karl Dilcher, Frithjof Lutscher, Nilima Nigam, and Keith Taylor. The series publishes high-impact works across the breadth of mathematics and its applications. Books in this series will appeal to all mathematicians, students and established researchers. The series replaces the CMS Books in Mathematics series that successfully published over 45 volumes in 20 years.



CMS
SMC



CAIMS
SCMAI

David E. Stewart

Numerical Analysis: A Graduate Course

 Springer

David E. Stewart
Department of Mathematics
University of Iowa
Iowa, IA, USA

ISSN 2730-650X ISSN 2730-6518 (electronic)
CMS/CAIMS Books in Mathematics
ISBN 978-3-031-08120-0 ISBN 978-3-031-08121-7 (eBook)
<https://doi.org/10.1007/978-3-031-08121-7>

Mathematics Subject Classification: 65-01

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Numerical analysis has advanced greatly since it began as a way of creating methods to approximate answers to mathematical questions. This book aims to bring students closer to the frontier regarding the numerical methods that are used. But this book is not only about newer, as well as “classical”, numerical methods. Rather the aim is to also explain how and why they work, or fail to work. This means that there is a significant amount of theory to be understood. Simple analyses can result in methods that usually work, but can then fail in certain circumstances, sometimes catastrophically. The causes of success of a numerical algorithm and its failure are both important. Without understanding the underlying theory, the reasons for a method’s success and failure remain mysterious, and we do not have a means to determine how to fix the problem(s).

In this way, numerical analysis is a dialectic¹ between practice and theory; the practice being computation and programming, and the theory being mathematical analysis based on our model of computation. While we do indeed prove theorems in numerical analysis, the assumptions made in these theorems may not hold in many situations. Also, the conclusions may involve statements of the form “as n goes to infinity” (or “as h goes to zero”) while in actual computations n might not be especially large (or h especially small). Numerical analysis will sometimes ignore errors that we know exist (like roundoff error in the analysis of a method for solving differential equations). This is usually based on an understanding that some sources of errors are insignificant in a particular situation. Of course, there will be situations where roundoff error should be considered in the solution of differential equations, but only if the step size becomes unusually small. In that case, new analyses, and even new methods, may be necessary.

¹*Dialectic* is a dialog between a claim (a *thesis*) and counter-claims (an *antithesis*) hopefully leading to a new understanding (a *synthesis*) that incorporates both the original claim and the counter-claims. The synthesis is expected to give further understanding, but will itself eventually meet counter-claims.

Inspiration for this book has been found in the books of Atkinson and Han [11], Atkinson [13], Sauer [228], and Stoer and Bulirsch [241]. However, we wish to include current issues and interests that were not addressed in these books.

We aim to present numerical methods and their analysis in the context of modern applications and models. For example, the standard asymptotic error analysis of differential equations gives no advantage to implicit methods, which have a much larger computational cost. But for “stiff” problems there is a clear, and often decisive, advantage to implicit methods. While “stiffness” can be hard to quantify, it is also common in applications. We also wish to emphasize multivariate problems alongside single-variable problems: multivariate problems are crucial for partial differential equations, optimization, and integration over high-dimensional spaces. We deal with issues regarding randomness, including pseudo-random number generators, stochastic differential equations, and randomized algorithms. Stochastic differential equations meet a need for incorporating randomness into differential equations. High-dimensional integration is needed for studying questions and models in data science and simulation.

To summarize, I believe numerical analysis must be understood and taught in the context of applications, not simply as a discipline devoted solely to its own internal issues. Rather, these internal issues arise from understanding the common ground between analysis and applications. This is where the future of the discipline lies.

I would like to thank the many people who have been supportive of this effort, or contributed to it in some way. I would like to thank (in alphabetical order) Jeongho Ahn, Kendall Atkinson, Bruce Ayati, Ibrahim Emirahmetoglu, Koung-Hee Leem, Paul Muhly, Ricardo Rosado-Ortiz, and Xueyu Zhu. My wife, Suely Oliveira, has a special thanks for both encouraging this project and having the patience for me to see it through. Finally, I would like to thank the staff at Springer for their interest and support for this book, most especially Donna Chernyk.

How to Use This Book:

Numerical analysis is a combination of theory and practice. The theory is a mixture of calculus and analysis with some algorithm analysis thrown in. Practice is computation and programming. The algorithms in the book are shown as pseudo-code. Working code for MATLAB and/or Julia can be found at <https://github.com/destewart2022/NumerAnal-Gradbook>. The exercises are intended to develop both, and students need practice at both.

Like most intellectual disciplines, numerical analysis is more a spiral than a straight line. There is no linear ordering of the topics that makes complete sense. Thus teaching from this book should not be a matter of starting at one cover and ending at the other. In any case, there is probably too much material and an instructor must of necessity choose what they wish to emphasize. Matrix

computations are foundational, but even just focusing on this could easily take a semester or more. Differential equations arise in many, many applications, but the technical issues in partial differential equations can be daunting. The treatment here aims to be accessible without “dumbing down” the material. Students in data science may want to focus on optimization, high-dimensional approximation, and high-dimensional integration. Randomness finds its way into many applications, whether we wish it or not. So, here is a plan that you might consider when you first teach from this book:

- Chapter 1: A little on computing machinery to get started, floating point arithmetic, norms for vectors and matrices, and at least one-variable Taylor series with remainder.
- Chapter 2: LU factorization for linear systems, linear least squares via the normal equations and the QR factorization as a black box. Eigenvalues can wait until later.
- Chapter 3: Bisection, fixed-point, Newton, and secant methods are the foundation; guarded multivariate Newton and one-variable hybrid methods give good examples of how to modify algorithms for better reliability.
- Chapter 4: Polynomial interpolation is so central that you need to cover this well, including the error formula and the Runge phenomenon; the Weierstrass and Jackson theorems (without proof) give a sense of rate of convergence. Cubic splines give useful alternatives to plain polynomials. Lebesgue numbers may appear abstract, but give a good sense of the reliability of interpolation schemes. Radial basis functions give an entry into high-dimensional approximation.
- Chapter 5: Simple ideas can go a long way, but “integrate the interpolant” is a central idea; multivariate integration is also valuable here if you want to use it for partial differential equations.
- Chapter 6: Basic methods for solving ordinary differential equations are still very useful, although the revolution brought about by John Butcher’s approach to Runge–Kutta methods is worth a look—if you have time. Partial differential equations need some more set-up time, but are worthwhile for more advanced students, or a second time around. The scale of the problems for partial differential equations means that you should point your students back to Chapter 2 on how to solve large linear systems.
- Chapter 7: Randomness is important, and some statistical computation using SVDs may be a doorway to these issues. Random algorithms are also very important, but often involve advanced ideas.
- Chapter 8: Optimization has become more important in a number of areas, and including it is an option to consider. If your students want to do machine learning, some outline of the algorithms is available here.

A second course could be focused on specific issues. A machine learning focus could include iterative methods and SVDs for matrix computations in Chapter 2, radial basis functions from Chapter 4, high-dimensional integration from Chapter 5, methods for large-scale optimization from Chapter 8, rounded out with some

analysis of random algorithms from Chapter 7. A simulation-based course would focus on approximation and interpolation in two or three dimensions from Chapter 4, multi-dimensional integration from Chapter 5, much of the material from Chapter 6 on differential equations, both ordinary and partial. Uncertainty quantification could be served by starting with Chapter 7, progressing to iterative methods for large linear systems in Chapter 2, radial basis functions in Chapter 4, perhaps some partial differential equations in Chapter 6 and optimization in Chapter 8.

Notes on Notation:

Scalars are usually denoted by lower case italic letters (such as x, y, z, α, β) while vectors are usually denoted by lower case bold letters (such as $\mathbf{x}, \mathbf{y}, \mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}$). Matrices are usually denoted by upper case italic letters (X, Y, A, B). Entries of a vector \mathbf{x} are scalars, and so denoted x_i ; entries of a matrix A are denoted a_{ij} . Matrices and vectors usually have indexes that are integers $i = 1, 2, \dots, m, j = 1, 2, \dots, n$ for an $m \times n$ matrix. Sometimes it is convenient to have index sets that are different, so a matrix $A = [a_{ij} | i \in R, j \in C]$ can have row index set R and column index set C , and $\mathbf{x} = [x_i | i \in C]$ has index set C . This means $A\mathbf{x} = [\sum_{j \in C} a_{ij}x_j | i \in R]$ is the matrix–vector product. Just as A^T is used to denote the transpose of A , A^{-T} is the inverse of A^T , which is also the transpose of A^{-1} .

Functions are usually introduced by naming them. For example, the squaring function can be introduced as $q(x) = x^2$. Functions of several variables can be introduced as $f(x, y) = x^2 + y^2$ or using vectors as $f(\mathbf{x}) = \mathbf{x}^T\mathbf{x}$. Anonymous functions can be introduced as $\mathbf{x} \mapsto \mathbf{x}^T\mathbf{x}$.

Pseudo-code uses “ \leftarrow ” to assign a value to a variable (such as $x \leftarrow y$ assigns the value of y to x) while “ $=$ ” tests for equality (where $x = y$ returns *true* if x and y have the same value).

In some occasions, “ $:=$ ” is used to define a quantity of function where using “ $=$ ” might be ambiguous. The sets $\mathbb{R}, \mathbb{C},$ and \mathbb{Z} are understood to be the set of real numbers, the set of complex numbers, and the set of integers, respectively.

Iowa, USA

David E. Stewart

Contents

1	Basics of Numerical Computation	1
1.1	How Computers Work	1
1.1.1	The Central Processing Unit	2
1.1.2	Code and Data	2
1.1.3	On Being Correct	6
1.1.4	On Being Efficient	7
1.1.5	Recursive Algorithms and Induction	9
1.1.6	Working in Groups: Parallel Computing	11
1.1.7	BLAS and LAPACK	14
	Exercises	14
1.2	Programming Languages	18
1.2.1	MATLAB TM	18
1.2.2	Julia	19
1.2.3	Python	20
1.2.4	C/C++ and Java	20
1.2.5	Fortran	21
	Exercises	22
1.3	Floating Point Arithmetic	23
1.3.1	The IEEE Standards	23
1.3.2	Correctly Rounded Arithmetic	26
1.3.3	Future of Floating Point Arithmetic	29
	Exercises	32
1.4	When Things Go Wrong	33
1.4.1	Underflow and Overflow	33
1.4.2	Subtracting Nearly Equal Quantities	36
1.4.3	Numerical Instability	40
1.4.4	Adding Many Numbers	41
	Exercises	43

1.5	Measuring: Norms	44
1.5.1	What Is a Norm?	44
1.5.2	Norms of Functions	46
	Exercises	47
1.6	Taylor Series and Taylor Polynomials	47
1.6.1	Taylor Series in One Variable	48
1.6.2	Taylor Series and Polynomials in More than One Variable	50
1.6.3	Vector-Valued Functions	53
	Exercises	54
	Project	55
2	Computing with Matrices and Vectors	57
2.1	Solving Linear Systems	57
2.1.1	Gaussian Elimination	58
2.1.2	LU Factorization	61
2.1.3	Errors in Solving Linear Systems	63
2.1.4	Pivoting and $PA = LU$	71
2.1.5	Variants of LU Factorization	77
	Exercises	85
2.2	Least Squares Problems	87
2.2.1	The Normal Equations	88
2.2.2	QR Factorization	96
	Exercises	105
2.3	Sparse Matrices	106
2.3.1	Tridiagonal Matrices	106
2.3.2	Data Structures for Sparse Matrices	109
2.3.3	Graph Models of Sparse Factorization	111
2.3.4	Unsymmetric Factorizations	117
	Exercises	117
2.4	Iterations	119
2.4.1	Classical Iterations	119
2.4.2	Conjugate Gradients and Krylov Subspaces	126
2.4.3	Non-symmetric Krylov Subspace Methods	134
	Exercises	141
2.5	Eigenvalues and Eigenvectors	143
2.5.1	The Power Method & Google	143
2.5.2	Schur Decomposition	151
2.5.3	The QR Algorithm	158
2.5.4	Singular Value Decomposition	169
2.5.5	The Lanczos and Arnoldi Methods	175
	Exercises	177

- 3 Solving nonlinear equations** 181
 - 3.1 Bisection method 181
 - 3.1.1 Convergence 182
 - 3.1.2 Robustness and reliability 183
 - Exercises 184
 - 3.2 Fixed-point iteration 185
 - 3.2.1 Convergence 186
 - 3.2.2 Robustness and reliability 188
 - 3.2.3 Multivariate fixed-point iterations 189
 - Exercises 191
 - 3.3 Newton’s method 193
 - 3.3.1 Convergence of Newton’s method 194
 - 3.3.2 Reliability of Newton’s method 196
 - 3.3.3 Variant: Guarded Newton method 197
 - 3.3.4 Variant: Multivariate Newton method 200
 - Exercises 203
 - 3.4 Secant and hybrid methods 205
 - 3.4.1 Convenience: Secant method 205
 - 3.4.2 Regula Falsi 208
 - 3.4.3 Hybrid methods: Dekker’s and Brent’s methods 210
 - Exercises 212
 - 3.5 Continuation methods 215
 - 3.5.1 Following paths 215
 - 3.5.2 Numerical methods to follow paths 218
 - Exercises 223
 - Project 223
- 4 Approximations and Interpolation** 227
 - 4.1 Interpolation—Polynomials 227
 - 4.1.1 Polynomial Interpolation in One Variable 228
 - 4.1.2 Lebesgue Numbers and Reliability 249
 - Exercises 254
 - 4.2 Interpolation—Splines 256
 - 4.2.1 Cubic Splines 257
 - 4.2.2 Higher Order Splines in One Variable 266
 - Exercises 266
 - 4.3 Interpolation—Triangles and Triangulations 268
 - 4.3.1 Interpolation over Triangles 268
 - 4.3.2 Interpolation over Triangulations 278
 - 4.3.3 \triangle Approximation Error over Triangulations 283
 - 4.3.4 Creating Triangulations 286
 - Exercises 289
 - 4.4 Interpolation—Radial Basis Functions 291
 - Exercises 294

- 4.5 Approximating Functions by Polynomials 295
 - 4.5.1 Weierstrass’ Theorem 296
 - 4.5.2 Jackson’s Theorem 297
 - 4.5.3 Approximating Functions on Rectangles and Cubes 298
- 4.6 Seeking the Best—Minimax Approximation 299
 - 4.6.1 Chebyshev’s Equi-oscillation Theorem 299
 - 4.6.2 Chebyshev Polynomials and Interpolation 304
 - 4.6.3 Remez Algorithm 306
 - 4.6.4 Minimax Approximation in Higher Dimensions 307
 - Exercises 308
- 4.7 Seeking the Best—Least Squares 311
 - 4.7.1 Solving Least Squares 311
 - 4.7.2 Orthogonal Polynomials 314
 - 4.7.3 Trigonometric Polynomials and Fourier Series 316
 - 4.7.4 Chebyshev Expansions 321
 - Exercises 322
- Project 323
- 5 Integration and Differentiation 325**
 - 5.1 Integration via Interpolation 325
 - 5.1.1 Rectangle, Trapezoidal and Simpson’s Rules 325
 - 5.1.2 Newton–Cotes Methods 333
 - 5.1.3 Product Integration Methods 336
 - 5.1.4 Extrapolation 338
 - Exercises 341
 - 5.2 Gaussian Quadrature 343
 - 5.2.1 Orthogonal Polynomials Reprise 343
 - 5.2.2 Orthogonal Polynomials and Integration 344
 - 5.2.3 Why the Weights are Positive 346
 - Exercises 347
 - 5.3 Multidimensional Integration 348
 - 5.3.1 Tensor Product Methods 348
 - 5.3.2 Lagrange Integration Methods 349
 - 5.3.3 Symmetries and Integration 350
 - 5.3.4 Triangles and Tetrahedra 351
 - Exercises 354
 - 5.4 High-Dimensional Integration 356
 - 5.4.1 Monte Carlo Integration 356
 - 5.4.2 Quasi-Monte Carlo Methods 364
 - Exercises 371
 - 5.5 Numerical Differentiation 372
 - 5.5.1 Discrete Derivative Approximations 373
 - 5.5.2 Automatic Differentiation 378
 - Exercises 383

6	Differential Equations	385
6.1	Ordinary Differential Equations — Initial Value Problems	385
6.1.1	Basic Theory	386
6.1.2	Euler’s Method and Its Analysis	391
6.1.3	Improving on Euler: Trapezoidal, Midpoint, and Heun	395
6.1.4	Runge–Kutta Methods	397
6.1.5	Multistep Methods	409
6.1.6	Stability and Implicit Methods	412
6.1.7	Practical Aspects of Implicit Methods	423
6.1.8	Error Estimates and Adaptive Methods	428
6.1.9	Differential Algebraic Equations (DAEs)	432
	Exercises	438
6.2	Ordinary Differential Equations—Boundary Value Problems	440
6.2.1	Shooting Methods	442
6.2.2	Multiple Shooting	444
6.2.3	Finite Difference Approximations	445
	Exercises	446
6.3	Partial Differential Equations—Elliptic Problems	448
6.3.1	Finite Difference Approximations	450
6.3.2	Galerkin Method	457
6.3.3	Handling Boundary Conditions	470
6.3.4	Convection—Going with the Flow	474
6.3.5	Higher Order Problems	475
	Exercises	476
6.4	Partial Differential Equations—Diffusion and Waves	478
6.4.1	Method of Lines	479
	Exercises	484
	Projects	487
7	Randomness	489
7.1	Probabilities and Expectations	489
7.1.1	Random Events and Random Variables	489
7.1.2	Expectation and Variance	493
7.1.3	Averages	496
	Exercises	497
7.2	Pseudo-Random Number Generators	497
7.2.1	The Arithmetical Generation of Random Digits	499
7.2.2	Modern Pseudo-Random Number Generators	502
7.2.3	Generating Samples from Other Distributions	505
7.2.4	Parallel Generators	507
	Exercises	509

7.3	Statistics	509
7.3.1	Averages and Variances	510
7.3.2	Regression and Curve Fitting	511
7.3.3	Hypothesis Testing	513
	Exercises	516
7.4	Random Algorithms	517
7.4.1	Random Choices	517
7.4.2	Monte Carlo Algorithms and Markov Chains	518
	Exercises	522
7.5	Stochastic Differential Equations	524
7.5.1	Wiener Processes	525
7.5.2	Itô Stochastic Differential Equations	527
7.5.3	Stratonovich Integrals and Differential Equations	529
7.5.4	Euler–Maruyama Method	531
7.5.5	Higher Order Methods for Stochastic Differential Equations	532
	Exercises	534
	Project	536
8	Optimization	537
8.1	Basics of Optimization	537
8.1.1	Existence of Minimizers	537
8.1.2	Necessary Conditions for Local Minimizers	539
8.1.3	Lagrange Multipliers and Equality-Constrained Optimization	542
	Exercises	547
8.2	Convex and Non-convex	548
8.2.1	Convex Functions	548
8.2.2	Convex Sets	551
	Exercises	553
8.3	Gradient Descent and Variants	553
8.3.1	Gradient Descent	554
8.3.2	Line Searches	557
8.3.3	Convergence	562
8.3.4	Stochastic Gradient Method	569
8.3.5	Simulated Annealing	574
	Exercises	577
8.4	Second Derivatives and Newton’s Method	578
	Exercises	581
8.5	Conjugate Gradient and Quasi-Newton Methods	583
8.5.1	Conjugate Gradients for Optimization	583
8.5.2	Variants on the Conjugate Gradient Method	586
8.5.3	Quasi-Newton Methods	587
	Exercises	589

8.6	Constrained Optimization	590
8.6.1	Equality Constrained Optimization	591
8.6.2	Inequality Constrained Optimization	592
	Exercises	597
	Project	599
Appendix A: What You Need from Analysis		601
References		611
Index		623