

**TEXTS IN COMPUTATIONAL SCIENCE
AND ENGINEERING**

13

Mark H. Holmes

Introduction to Scientific Computing and Data Analysis

Editorial Board

T. J. Barth

M. Griebel

D. E. Keyes

R. M. Nieminen

D. Roose

T. Schlick

 **Springer**

Mark H. Holmes

Introduction to Scientific Computing and Data Analysis



Springer

Mark H. Holmes
Department of Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY, USA

ISSN 1611-0994 ISSN 2197-179X (electronic)
Texts in Computational Science and Engineering
ISBN 978-3-319-30254-6 ISBN 978-3-319-30256-0 (eBook)
DOI 10.1007/978-3-319-30256-0

Library of Congress Control Number: 2016935931

Mathematics Subject Classification (2010): 65-01, 15-01, 49Mxx, 49Sxx, 65D05, 65D07, 65D25, 65D30, 65D32, 65Fxx, 65Hxx, 65K10, 65L05, 65L12, 65Zxx

© Springer International Publishing Switzerland 2016

This work is subject to copyright. All rights are reserved 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, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG Switzerland

Editors

Timothy J. Barth

Michael Griebel

David E. Keyes

Risto M. Nieminen

Dirk Roose

Tamar Schlick

Preface

The objective of this text is easy to state, and it is to investigate ways to use a computer to solve various mathematical problems. One of the challenges for those learning this material is that it involves a nonlinear combination of mathematical analysis and nitty-gritty computer programming. Texts vary considerably in how they balance these two aspects of the subject. You can see this in the brief history of the subject given in Figure 1 (which is an example of what is called an ngram plot). According to this plot, the earlier books concentrated more on the analysis (theory). In the early 1970s this changed, and there was more of an emphasis on methods (which generally means much less theory), and these continue to dominate the area today. However, the 1980s saw the advent of scientific computing books, which combine theory and programming, and you can see a subsequent decline in the other two types of books when this occurred. This text falls within this latter group.

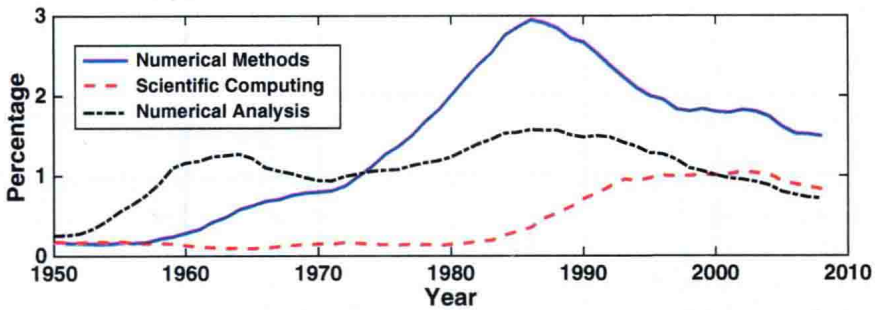


Figure 1 Historical record according to Google. The values are the number of instances that the expression appeared in a published book in the respective year, expressed as a percentage for that year, times 10^5 [Michel et al., 2011].

There are two important threads running through the text. One concerns understanding the mathematical problem that is being solved. As an example, when using Newton's method to solve $f(x) = 0$, the usual statement is that it will work if you guess a starting value close to the solution. It is important to know how to determine good starting points and, perhaps even more importantly, whether the problem being solved even has a solution. Consequently, when deriving Newton's method, and others like it, an effort is made to explain how to fairly easily answer these questions.

The second theme is the importance in scientific computing of having a solid grasp of the theory underlying the methods being used. A computer has the unfortunate ability to produce answers even if the methods used to find the solution are completely wrong. Consequently, it is essential to have an understanding of how the method works and how the error in the computation depends on the method being used.

Needless to say, it is also important to be able to code these methods and in the process be able to adapt them to the particular problem being solved. There is considerable room for interpretation on what this means. To explain, in terms of computing languages, the current favorites are MATLAB and Python. Using the commands they provide, a text such as this one becomes more of a user's manual, reducing the entire book down to a few commands. For example, with MATLAB, this book (as well as most others in this area) can be replaced with the following commands:

Chapter 1: `eps`
Chapter 2: `fzero(@f,x0)`
Chapter 3: `A\b`
Chapter 4: `eig(A)`
Chapter 5: `polyfit(x,y,n)`
Chapter 6: `integral(@f,a,b)`
Chapter 7: `ode45(@f,tspan,y0)`
Chapter 8: `fminsearch(@fun,x0)`
Chapter 9: `svd(A)`

Certainly this statement qualifies as hyperbole, and, as an example, Chapters 4 and 5 should probably have two commands listed. The other extreme is to write all of the methods from scratch, something that was expected of students in the early days of computing. In the end, the level of coding depends on what the learning outcomes are for the course and the background and computing prerequisites required for the course.

Many of the topics included are typical of what are found in an upper-division scientific computing course. There are also notable additions. This includes material related to data analysis, as well as variational methods and derivative-free minimization methods. Moreover, there are differences related to emphasis. An example here concerns the preeminent role matrix factorizations play in numerical linear algebra, and this is made evident in the development of the material.

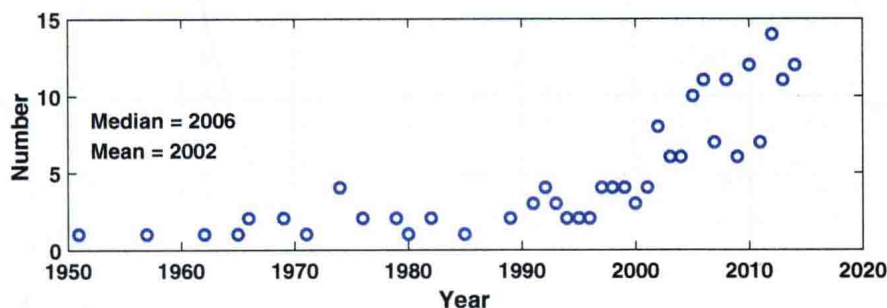


Figure 2 The number of references in this book, after 1950, as a function of the year they were published.

The coverage of any particular topic is not exhaustive, but intended to introduce the basic ideas. For this reason, numerous references are provided for those who might be interested in further study, and many of these are from the current research literature. To quantify this statement, a code was written that reads the *tex.bbl* file containing the references for this text and then uses MATLAB to plot the number as a function of the year published. The result is Figure 2, and it shows that approximately half of the references were published in the last ten years. By the way, in terms of data generation and plotting, Figure 1 was produced by writing a code which reads the html source code for the ngram web page and then uses MATLAB to produce the plot.

The MATLAB codes used to produce almost every figure, and table with numerical output, in this text are available from the author's web site as well as from SpringerLink. In other words, the MATLAB codes for all of the methods considered, and the examples used, are available. These can be used as a learning tool. This also goes to the importance in computational-based research, and education, of providing open source to guarantee the correctness and reproducibility of the work. Some interesting comments on this can be found in Morin et al. [2012] and Peng [2011].

The prerequisites depend on which chapters are covered, but the typical two-year lower-division mathematics program (consisting of calculus, matrix algebra, and differential equations) should be sufficient for the entire text. However, one topic plays an oversized role in this subject, and this is Taylor's theorem. This also tends to be the topic that students had the most trouble with in calculus. For this reason, an appendix is included that reviews some of the more pertinent aspects of Taylor's theorem. It should also be pointed out that there are numerous theorems in the text, as well as an outline of the proof for many of them. These should be read with care because they contain information that is useful when testing the code that implements the respective method (i.e., they provide one of the essential ways we will have to make sure the computed results are actually correct).

I would like to thank the reviewers of an early draft of the book, who made several very constructive suggestions to improve the text. Also, as usual, I would like to thank those who developed and have maintained TeXShop, a free and very good TeX previewer.

Troy, NY, USA
January 2016

Mark H. Holmes

Contents

Preface	v
1 Introduction to Scientific Computing	1
1.1 Unexpected Results	1
1.2 Floating-Point Number System	5
1.2.1 Normal Floats	5
1.2.2 Machine Epsilon	7
1.2.3 Rounding	9
1.2.4 Nonnormal Floats	9
1.2.5 Flops	11
1.2.6 Functions	12
1.3 Arbitrary-Precision Arithmetic	12
1.4 Explaining, and Possibly Fixing, the Unexpected Results ...	13
1.5 Error and Accuracy	18
1.5.1 Test Cases	20
1.5.2 Over-Computing?	21
2 Solving A Nonlinear Equation	31
2.1 Examples	31
2.1.1 Physical	31
2.1.2 Mathematical	33
2.2 The Problem to Solve	35
2.3 Bisection Method	35
2.4 Newton's Method	40
2.4.1 Order of Convergence	45
2.4.2 Failure	45
2.4.3 Some Theory	46
2.5 Secant Method	50
2.5.1 Some Theory	53

2.6	Other Ideas	54
2.6.1	Is Newton's Method Really Newton's Method?	55
3	Matrix Equations	71
3.1	An Example	71
3.2	Finding L and U	73
3.2.1	What Matrices Have an LU Factorization?	74
3.2.2	Factoring $n \times n$ Matrices	76
3.2.3	Pivoting Strategies	77
3.3	LU and Gaussian Elimination	78
3.4	LU Method: Summary	80
3.5	Vector and Matrix Norms	85
3.5.1	Matrix Norms	87
3.6	Error and Residual	89
3.6.1	Significant Digits	90
3.6.2	The Condition Number	91
3.6.3	A Heuristic	94
3.7	Positive Definite Matrices	95
3.7.1	Cholesky Factorization	98
3.8	Tri-Diagonal Matrices	100
3.9	Sparse Matrices	102
3.10	Nonlinear Systems	102
3.11	Some Additional Ideas	106
3.11.1	Yogi Berra and Perturbation Theory	106
3.11.2	Fixing an Ill-Conditioned Matrix	106
3.11.3	Insightful Observations About the Condition Number .	107
3.11.4	Faster than LU?	109
3.11.5	Historical Comparisons	110
4	Eigenvalue Problems	121
4.1	Power Method	125
4.1.1	General Formulation	130
4.2	Extensions of the Power Method	133
4.2.1	Inverse Power Method	133
4.2.2	Inverse Iteration	134
4.2.3	Rayleigh Quotient Iteration	137
4.3	Calculating Multiple Eigenvalues	139
4.3.1	Orthogonal Iteration	140
4.3.2	QR Factorization	146
4.3.3	The QR Method	148
4.3.4	Are the Computed Values Correct?	151
4.4	Applications	152
4.4.1	Natural Frequencies	153
4.4.2	Graphs and Networks	156

4.5	Singular Value Decomposition	158
4.5.1	Derivation of the Singular Value Decomposition	160
4.5.2	Summary of the Singular Value Decomposition	162
4.5.3	Application: Image Compression	166
5	Interpolation	183
5.1	Information from Data	183
5.2	Global Polynomial Interpolation	185
5.2.1	Direct Approach	185
5.2.2	Lagrange Approach	187
5.2.3	Runge's Function	189
5.3	Piecewise Linear Interpolation	190
5.4	Piecewise Cubic Interpolation	194
5.4.1	Cubic B-Splines	197
5.5	Function Interpolation	202
5.5.1	Global Polynomial Interpolation	202
5.5.2	Piecewise Linear Interpolation	205
5.5.3	Cubic Splines	207
5.5.4	Chebyshev Interpolation	209
5.5.5	Chebyshev Versus Cubic Splines	214
5.5.6	Other Ideas	216
5.6	Questions and Additional Comments	217
6	Numerical Integration	231
6.1	Introduction	231
6.2	The Definition from Calculus	232
6.2.1	Midpoint Rule	234
6.3	Methods Based on Polynomial Interpolation	237
6.3.1	Trapezoidal Rule	238
6.3.2	Simpson's Rule	241
6.3.3	Cubic Splines	244
6.3.4	Other Interpolation Ideas	246
6.4	Methods Based on Precision	247
6.4.1	1-Point Gaussian Rule	248
6.4.2	2-Point Gaussian Rule	249
6.4.3	Error Formulas	250
6.4.4	General Case	253
6.5	Romberg Integration	255
6.5.1	Computing Using Romberg	256
6.6	Adaptive Quadrature	258
6.7	Other Ideas	263
6.8	Epilogue	263

7	Initial Value Problems	275
7.1	Examples of IVPs	275
7.1.1	Radioactive Decay	275
7.1.2	Logistic Equation	276
7.2	Numerical Differentiation	277
7.2.1	Using t_{j+2} , t_{j+1} , and t_j	278
7.2.2	Using t_{j+1} and t_{j-1}	279
7.2.3	Higher Derivatives	281
7.2.4	Interpolation	283
7.3	IVP Methods Using Numerical Differentiation	283
7.3.1	The Five Steps	283
7.3.2	Error	290
7.3.3	Additional Difference Methods	292
7.3.4	Extensions	295
7.4	IVP Methods Using Numerical Integration	295
7.5	Runge-Kutta Methods	298
7.5.1	RK2	299
7.5.2	RK4	301
7.5.3	Stability	304
7.5.4	RK-n	304
7.6	Solving Systems of IVPs	304
7.6.1	Examples	305
7.6.2	Simple Approach	306
7.6.3	Component Approach and Symplectic Methods	308
7.7	Some Additional Questions and Ideas	310
7.7.1	RK4: Why Use Simpson's Rule?	313
8	Optimization	327
8.1	Introduction	327
8.2	Regression: Introduction	331
8.2.1	Model Function	331
8.2.2	Error Function	332
8.3	Linear Least Squares	335
8.3.1	Two Parameters	335
8.3.2	General Case	337
8.3.3	Other Error Functions	340
8.4	Nonlinear Regression	344
8.4.1	Transforming to Linear Regression	345
8.5	Descent Methods: Introduction	347
8.5.1	Descent Directions	349
8.6	Solving Linear Systems	350
8.6.1	Basic Descent Algorithm for $\mathbf{A}\mathbf{v} = \mathbf{b}$	351
8.6.2	Method of Steepest Descents for $\mathbf{A}\mathbf{v} = \mathbf{b}$	352
8.6.3	Conjugate Gradient Method for $\mathbf{A}\mathbf{v} = \mathbf{b}$	354

8.7	Descent Methods: General Nonlinear Problem	360
8.7.1	Descent Direction	360
8.7.2	Line Search Problem	361
8.7.3	Examples	363
8.8	Minimization Without Differentiation	367
8.8.1	Examples	369
8.9	Variational Problems	372
8.9.1	Example: Minimum Potential Energy	372
8.9.2	Example: Brachistochrone Problem	375
8.9.3	Parting Comments	378
8.10	Global Minimum	379
9	Data Analysis	397
9.1	Introduction	397
9.2	Principal Component Analysis	397
9.2.1	Example: Word Length	398
9.2.2	Principal Component Decomposition	402
9.2.3	Scaling Factors	409
9.2.4	Application: Crime Data	410
9.2.5	Geometry and Data	414
9.2.6	Error	415
9.2.7	Parting Comments	417
9.3	Independent Component Analysis	418
9.3.1	Derivation of Method	420
9.3.2	Reduced Problem	423
9.3.3	Contrast Function	424
9.3.4	Summary of ICA	429
9.3.5	Application: Image Separation	430
9.4	Modal Data Analysis	432
9.4.1	Application: Google's Flu Data	434
9.4.2	Propagation Modes	435
9.4.3	Parting Comments	438
9.5	Fitting IVPs to Data	439
9.5.1	Logistic Equation	439
9.5.2	FitzHugh-Nagumo Equations	442
9.5.3	Mass-Spring-Dashpot System	442
9.5.4	Parting Comments	445
A	Taylor's Theorem	453
A.1	Useful Taylor Series for x Near Zero	455
A.2	Order Symbol and Truncation Error	456

B B-Splines 459

 B.1 Definition 459

 B.2 Plot 460

 B.3 Particular Values 460

 B.4 Derivatives 460

 B.5 Integrals 461

C Summary Tables 463

 Interpolation Methods 464

 Integration Methods 465

 Methods for IVPs 466

 Gradient Decent Methods 467

References 469

Index 483

Chapter 1

Introduction to Scientific Computing

This chapter provides a brief introduction to the floating-point number system used in most scientific and engineering applications. A few examples are given in the next section illustrating some of the challenges using finite precision arithmetic, but it is worth quoting Donald Knuth to get things started. If you are unfamiliar with him, he was instrumental in the development of the analysis of algorithms, and is the creator of TeX. Anyway, here are the relevant quotes [Knuth, 1997]:

“We don’t know how much of the computer’s answers to believe. Novice computer users solve this problem by implicitly trusting in the computer as an infallible authority; they tend to believe that all digits of a printed answer are significant. Disillusioned computer users have just the opposite approach; they are constantly afraid that their answers are almost meaningless.”

“every well-rounded programmer ought to have a knowledge of what goes on during the elementary steps of floating point arithmetic. This subject is not at all as trivial as most people think, and it involves a surprising amount of interesting information.”

One of the objectives in what follows is to help you from becoming disillusioned by identifying where problems can occur, and also to provide an appreciation for the difficulty of floating-point computation.

1.1 Unexpected Results

What follows are examples where the computed results are not what is expected. The reason for the problem is the same for each example. Namely, the finite precision arithmetic use by the computer generates errors that are

significant enough that they affect the final result. Note that the calculations to follow are from MATLAB, but the same, or similar, results are expected for any system using double precision arithmetic (this is defined in Section 1.2).

Example 1

Consider adding a series from largest to smallest

$$S(n) = 1 + \frac{1}{2} + \cdots + \frac{1}{n-1} + \frac{1}{n}, \quad (1.1)$$

and the same series added from smallest to largest

$$s(n) = \frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{2} + 1. \quad (1.2)$$

According to the usual rules of arithmetic these are equal. However, this does not necessarily happen when the sums are calculated with a computer. If one calculates $s(n)$ and $S(n)$, and then calculates the difference $S(n) - s(n)$, the values given in Table 1.1 are obtained. It is evident that for larger values of n , the two sums differ. The first question is why this happens, but there are other questions as well. For example, assuming both are incorrect, is it possible to determine which sum is closer to the exact result? ■

Example 2

Consider the function

$$y = (x - 1)^8. \quad (1.3)$$

If one expands this, the following is obtained

$$y = x^8 - 8x^7 + 28x^6 - 56x^5 + 70x^4 - 56x^3 + 28x^2 - 8x + 1. \quad (1.4)$$

n	$S(n) - s(n)$
10	0
100	-8.88e-16
1,000	2.66e-15
10,000	-3.73e-14
100,000	-7.28e-14
1,000,000	-7.83e-13

Table 1.1 Difference in partial sums for the harmonic series considered in Example 1. Note that $-8.9\text{e-}16 = -8.9 \times 10^{-16}$.

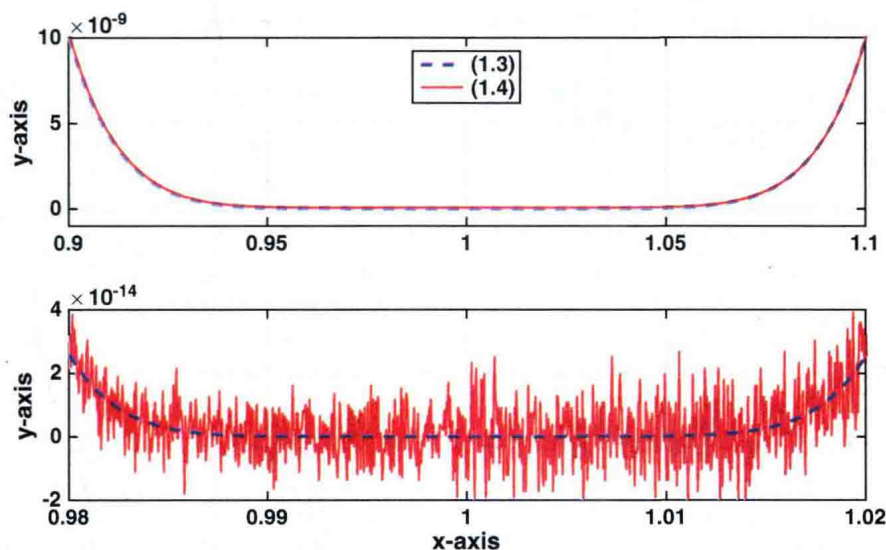


Figure 1.1 Plots of (1.4) and (1.3). Upper graph: the interval is $0.9 \leq x \leq 1.1$, and the two functions are so close that the curves are indistinguishable. Lower graph: the interval is $0.98 \leq x \leq 1.02$, and now they are not so close.

The expressions in (1.4) and (1.3) are equal and, given a value of x , either should be able to be used to evaluate the function. However, when evaluating them with a computer they do not necessarily produce the same values and that is shown in Figure 1.1. In the upper graph they do appear to agree, but that is certainly not true in the lower graph. The situation is even worse than the fact that the graphs differ. First, according to (1.3), y is never negative but according to the computer (1.4) violates this condition. Second, according to (1.3), y is symmetric about $x = 1$ but the computer claims (1.4) is not. ■

Example 3

As a third example, consider the function

$$y = \frac{\sqrt{16+k}-4}{k}. \quad (1.5)$$

This is plotted in Figure 1.2. According to l'Hospital's rule

$$\lim_{k \rightarrow 0} y = \frac{1}{8}.$$

The computer agrees with this result for k down to about 10^{-12} but for smaller values of k there is a problem. First, the function starts to oscillate