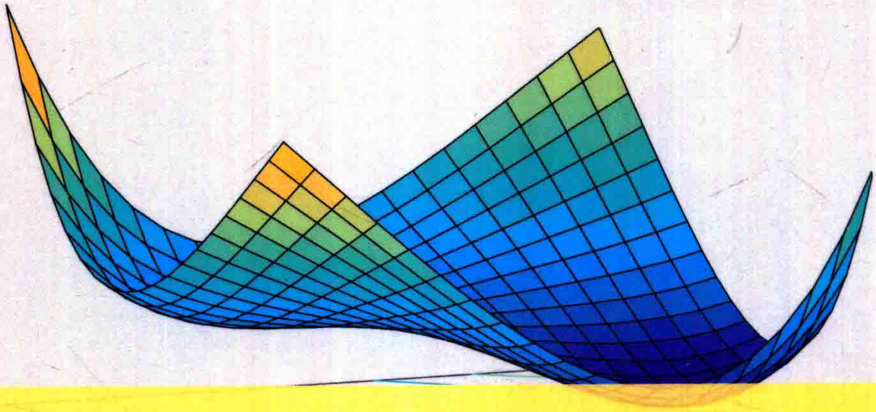

TEXTS IN COMPUTATIONAL SCIENCE AND ENGINEERING 13

Mark H. Holmes

Introduction to Scientific Computing and Data Analysis

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

Printed on acid-free paper

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

Texts in Computational Science and Engineering 13

Editors

Timothy J. Barth
Michael Griebel
David E. Keyes
Risto M. Nieminen
Dirk Roose
Tamar Schlick

More information about this series at http://www.springer.com/series/5151

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.

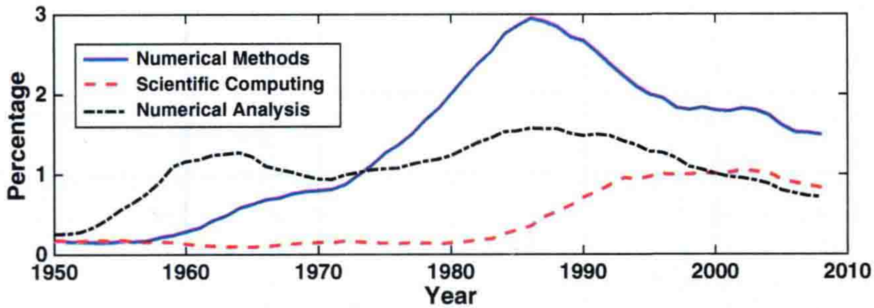

Figure 1 Historical record according to Google. The values are the number of instances that the expression appeared in a published book in the respective year, expressed as a percentage for that year, times 10^5 [Michel et al., 2011].

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.

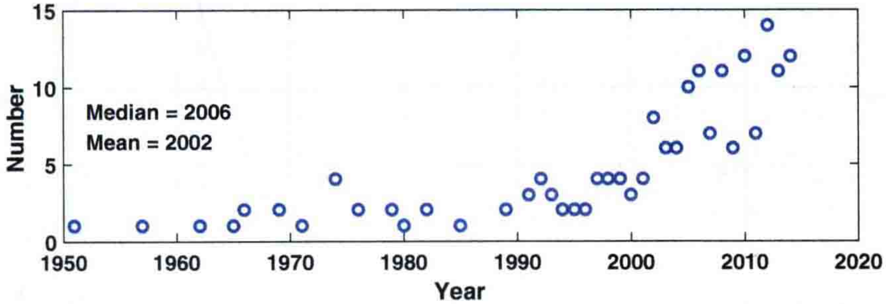

Figure 2 The number of references in this book, after 1950, as a function of the year they were published.

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

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

M.H. Holmes, *Introduction to Scientific Computing and Data Analysis*,
Texts in Computational Science and Engineering 13,
DOI 10.1007/978-3-319-30256-0_1

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

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

and the same series added from smallest to largest

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

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

$$y = (x-1)^8. \tag{1.3}$$

If one expands this, the following is obtained

$$y = x^8 - 8x^7 + 28x^6 - 56x^5 + 70x^4 - 56x^3 + 28x^2 - 8x + 1. \tag{1.4}$$

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 $-8.9\text{e}{-16} = -8.9 \times 10^{-16}$.

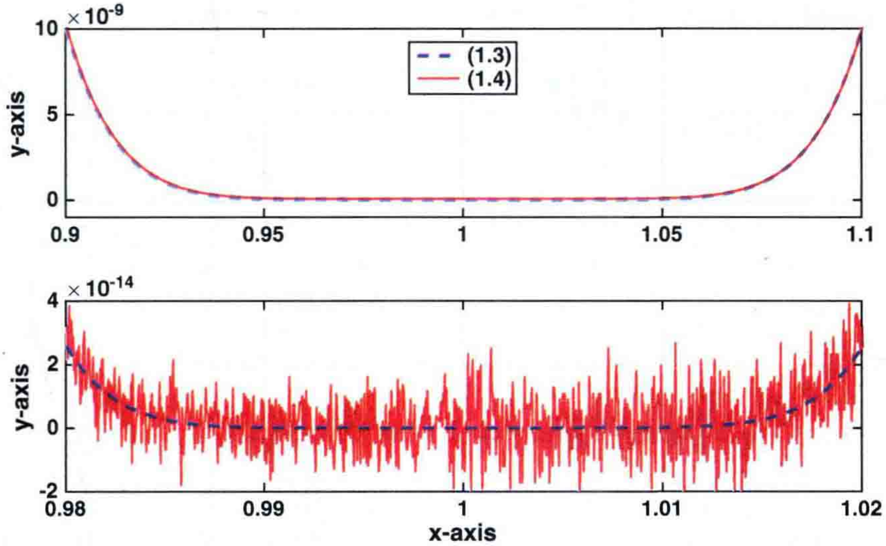

Figure 1.1 Plots of (1.4) and (1.3). Upper graph: the interval is $0.9 \leq x \leq 1.1$, and the two functions are so close that the curves are indistinguishable. Lower graph: the interval is $0.98 \leq x \leq 1.02$, and now they are not so close.

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

$$y = \frac{\sqrt{16+k} - 4}{k} . \tag{1.5}$$

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

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

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