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.

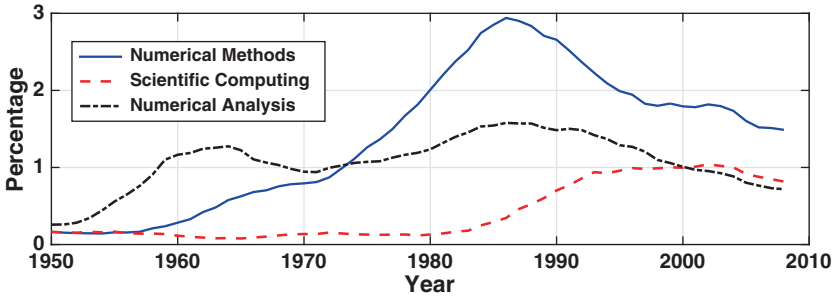

Figure 1 Historical record according to Google. The values are the number of instances that the expression appeared in a published book in the respective year, expressed as a percentage for that year, times 10^5 [Michel et al., 2011].

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.

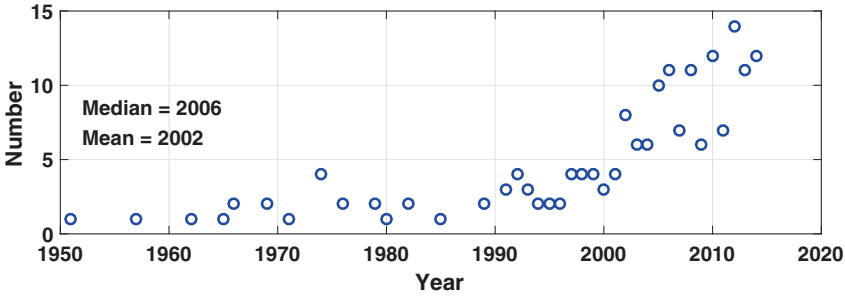

Figure 2 The number of references in this book, after 1950, as a function of the year they were published.

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 Mark H. Holmes
January 2016

http://www.springer.com/978-3-319-30254-6

Introduction to Scientific Computing and Data Analysis

Holmes, M.H.

2016, XIV, 497 p. 177 illus., 138 illus. in color.,
Hardcover
ISBN: 978-3-319-30254-6